Chopsticks: Fork-Free Two-Round Multi-Signatures from Non-Interactive Assumptions

Jiaxin Pan*[1] and Benedikt Wagner[2,3]

[1] NTNU – Norwegian University of Science and Technology, Trondheim, Norway
jiaxin.pan@ntnu.no
[2] CISPA Helmholtz Center for Information Security, Saarbrücken, Germany
benedikt.wagner@cispa.de
[3] Saarland University, Saarbrücken, Germany

Abstract. Multi-signatures have been drawing lots of attention in recent years, due to their applications in cryptocurrencies. Most early constructions require three-round signing, and recent constructions have managed to reduce the round complexity to two. However, their security proofs are mostly based on non-standard, interactive assumptions (e.g. one-more assumptions) and come with a huge security loss, due to multiple uses of rewinding (aka the Forking Lemma). This renders the quantitative guarantees given by the security proof useless.

In this work, we improve the state of the art by proposing two efficient two-round multi-signature schemes from the (standard, non-interactive) Decisional Diffie-Hellman (DDH) assumption. Both schemes are proven secure in the random oracle model without rewinding. We do not require any pairing either. Our first scheme supports key aggregation but has a security loss linear in the number of signing queries, and our second scheme is the *first* tightly secure construction.

A key ingredient in our constructions is a new homomorphic dual-mode commitment scheme for group elements, that allows to equivocate for messages of a certain structure. The definition and efficient construction of this commitment scheme is of independent interest.

Keywords. Multi-Signatures, Tightness, Forking Lemma, Commitment Scheme, Round Complexity

1 Introduction

A multi-signature scheme [24,5] allows N parties to jointly sign a message, where each party i holds an independent key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$. Recently, multi-signature schemes have been drawing new attention due to their applications in cryptocurrencies. In this setting, multiple parties share ownership of funds, and can use multi-signatures to sign transactions spending these funds. For details,

* Supported by the Research Council of Norway under Project No. 324235.

we refer to [8]. A trivial construction is that each signer i computes a signature σ_i using sk_i, and the final signature is $(\sigma_1, \ldots, \sigma_N)$. Yet, this trivial approach leads to large signature size. Motivated by this, cryptographers are proposing more sophisticated multi-signature schemes with interactive signing protocols to compress the signature size. In this work, we focus on concrete security of two-round multi-signature schemes.

Security Models. There are different models in which multi-signatures have been proposed and analyzed. Namely, schemes may require interactive key generation [31], or require that keys are verified and include a proof of possesion of the secret key [14,11]. Other schemes require to use a knowledge of secret key assumption [7,29]. Besides these models, the widely accepted model for multi-signatures nowadays is the so called *plain public key model*, introduced by Bellare and Neven in their seminal work [5]. In this model, each signer generates her key pair independently, and no knowledge assumption or proof of possession is needed. In this paper, we are interested in the plain public key model.

Concrete Security and Tightness. Cryptographic schemes are proven secure using reductions. To prove security of a scheme S, we transform any adversary $\mathcal{A}_{\mathsf{S}}$ against the security of S with success probability ϵ_{S} into a solver $\mathcal{A}_{\Pi}$ for some underlying hard problem Π with success probability ϵ_{Π}. Thereby, we establish a bound $\epsilon_{\mathsf{S}} \leq L \cdot \epsilon_{\Pi}$. We call L the *security loss*. Ideally, we want the underlying hardness assumption to be as standard as possible, since a more standard assumption gives us more confidence on the scheme's security. We also want the security loss as small as possible, since it relates the concrete security of our scheme to the hardness of the underlying computational problem. This is reflected when we use the security proof as a quantitative statement to derive concrete parameters for scheme S based on cryptanalytic results for the well-studied problem Π. Roughly speaking, to get κ bits of security for S, we have to guarantee $\kappa + \log L$ bits of security for Π. If L is large, or depends on choices of the adversary unknown at deployment time, instantiating the scheme in this way leads to prohibitively large parameters, or is not even possible. This motivates striving for a *tight reduction*, i.e. a reduction where L is a small constant. Tightness has been studied for many primitives, including standard digital signatures and related primitives, e.g., [26,6,28,22]. Unfortunately, most of existing multi-signature schemes are non-tight. Even worse, existing two-round multi-signature schemes have only non-tight reductions based on strong, non-standard assumptions.

Limitations of Existing Constructions. An overview of existing schemes (based on assumptions in cyclic groups) and their properties and security loss can be found in Table 1. In the plain public key model, Bellare and Neven [5] constructed a three-round multi-signature scheme (BN) based on the Discrete Logarithm Assumption (DLOG). Proving the security of this scheme relies on rewinding and uses the (general) Forking Lemma [5], which leads to a highly non-tight security bound. To improve this, Bellare and Neven introduced a second three-round construction ($\mathsf{BN}+$) tightly based on the Decisional Diffie-Hellman (DDH)

Assumption. Further works focus on key aggregation [30,8,16]. This feature allows to publicly compute a single aggregated key from a given list of public keys, which can later be used for verification. The key aggregation property saves bandwidth and is desirable in many applications. Notably, the three-round scheme Musig [30,8] can be seen as a variant of BN that supports key aggregation. The scheme is based on DLOG and a double forking technique is introduced for its analysis. This leads to a security bound of the form $\epsilon_S^4 \leq L \cdot \epsilon_\Pi$, which is useless in terms of concrete security. Using the Decisional Diffie-Hellman (DDH) assumption, a tightly secure variant Musig+ of Musig has been proposed in [16].

To further reduce round complexity, recent works focused on two-round constructions [32,4,2,11,13]. However, while achieving certain desirable properties (e.g. deterministic signing [33]) the proposed schemes have their drawbacks in terms of assumptions and concrete security. The scheme [33] makes use of heavy cryptographic machinery and is not comparable with others in terms of efficiency. Further, even in the more idealized models such as the algebraic group model, security proofs of most two-round schemes rely on non-standard interactive assumptions [32,11,4,2]. The only exceptions are [13,4,9]. A second drawback is the apparent need for (double) rewinding in the random oracle model [14,32,13,4,9]. While such security proofs show the absence of major structural attacks, concrete parameters are not supported by cryptanalytic evidence.

Our Goal. Motivated by the state of the art, we study whether interactive assumptions and rewinding techniques are necessary for two-round multi-signatures. If not, we want to construct a scheme without either of them. Ideally, our scheme comes with additional features such as key aggregation or a fully tight security proof. We summarize our central question as follows, which is of both practical and theoretical interest.

Can we construct two-round multi-signatures
from non-interactive pairing-free assumptions without the use of rewinding?

1.1 Our Contribution

Our work answers the above question in the affirmative. Our contributions are the *first* two multi-signature schemes that are two-round from a non-interactive assumption without using the Forking Lemma. Both of our schemes are proven secure in the random oracle model based on the DDH assumption. Concretely, we construct

1. a two-round multi-signature scheme with a security loss $O(Q_S)$ and key aggregation, where Q_S is the number of signing queries, and
2. the first two-round multi-signature scheme with a fully tight security proof

We compare our schemes with existing schemes in Table 1[4]. For roughly 128 bit security, our second scheme can be instantiated with standardized 128 bit secure

[4] We do not consider proofs in the (idealized) algebraic group model and do not list schemes that are not in the plain public key model.

curves, in contrast to all previous two-round schemes. For our first scheme, its proof is non-tight, but it does not rely on rewinding and has tighter security based on standard, non-interactive assumptions than other non-tight schemes (such as HBMS and Musig2). Hence, as long as the number of signing queries Q_S is less than $2^{192-128} = 2^{64}$, we can implement our first scheme with a standardized 192-bit secure curve to achieve 128-bit security, while this is not the case for HBMS and Musig2. We note that our schemes do not have some additional beneficial properties (e.g. having Schnorr-compatible signatures or supporting preprocessing) as in Musig2 [32]. We leave achieving these properties without rewinding as an interesting open problem.

Scheme	Assumption	Rounds	Key Aggregation	Loss
BN [5]	DLOG	3	✗	$O(Q_H/\epsilon)$
BN+ [5]	DDH	3	✗	$O(1)$
Musig [30,8]	DLOG	3	✓	$O(Q_H^3/\epsilon^3)$
Musig+ [16]	DDH	3	✓	$O(1)$
Musig2 [32]	AOMDL	2	✓	$O(Q_H^3/\epsilon^3)$
HBMS [4]	DLOG	2	✓	$O(Q_S^4 Q_H^3/\epsilon^3)$
Ours (Section 3.2)	DDH	2	✓	$O(Q_S)$
Ours (Section 3.3)	DDH	2	✗	$O(1)$

Table 1. Comparison of existing multi-signature schemes (top) in the random oracle model with our schemes (bottom). Here, Q_H, Q_S denote the number of random oracle and signing queries, respectively, ϵ denotes the advantage of an adversary against the scheme. The algebraic one-more discrete logarithm (AOMDL) assumption is a (stronger) interactive variant of DLOG.

A crucial building block for our construction is a special kind of DDH-based commitment scheme without pairings. Concretely, our commitment scheme has the following properties.

- It commits to pairs of group elements in a homomorphic way.
- It has a dual-mode property, i.e. indistinguishable keys in statistically hiding and statistically binding mode, with tight multi-key indistinguishability.
- The hiding mode offers a special form of equivocation trapdoor, which allows to open commitments to group elements output by the Honest-Verifier Zero-Knowledge (HVZK) simulator of Schnorr-like identification protocols.

Such a commitment scheme can be useful to construct other interactive signature variants, and we believe that this is of independent interest. In this paper, we construct the first commitment scheme satisfying the above properties simultaneously without using pairings. Our commitment scheme can be seen as an extension of the commitment scheme in [3][5]. Contrary to our scheme, the commitment

[5] Drijvers et al. [14] showed a flaw in the proof of the multi-signature scheme presented in [3], but it does not affect their commitment scheme.

scheme in [3] commits to single group elements and no statistically binding mode is shown, which makes it less desirable for our multi-signature constructions. Other previous commitment schemes either have no trapdoor property [19,20], or homomorphically commit to ring or field elements [21,35]. To the best of our knowledge, there is only a solution using pairings [18].

1.2 Concurrent Work

In a concurrent work (also at Eurocrypt 2023), Tessaro and Zhu [37] also presented (among other contributions) a new two-round multi-signature scheme. Both our work and theirs focus on avoiding interactive assumptions. However, while we additionally remove the security loss, Tessaro and Zhu concentrate on having a partially non-interactive scheme. That is, the first round of the signing protocol is independent of the message being signed. In a nutshell, they generalize Musig2 to linear function families. Then, under a suitable instantiation, the interactive assumption for Musig2 can be avoided. Similar to Musig2, the resulting scheme is partially non-interactive. Still, their scheme inherits the security loss of Musig2 due to (double) rewinding.

1.3 Technical Overview

We give an intuitive overview of our constructions and the challenges we solve.

Schnorr-Based Multi-Signatures. We start by recalling the basic template for multi-signatures based on the Schnorr identification scheme [36]. Let $\mathbb{G}$ be a group of prime order p with generator g. We explain the template using the vector space homomorphism $\mathsf{F} : x \mapsto g^x$ mapping from $\mathbb{Z}_p$ to $\mathbb{G}$, and write both domain and range additively. In a first approach to get a multi-signature scheme, we let each signer i with secret key sk_i sample a random $r_i \in \mathbb{Z}_p$, and send $R_i := \mathsf{F}(r_i)$ to all other signers. Then, an aggregated R is computed as $R = \sum_i R_i$. From this R, signers derive challenges c_i using a random oracle. Then, each signer computes a response $s_i = c_i\mathsf{sk}_i + r_i$ and sends this response. Finally, the signature contains R and the aggregated response $s = \sum_i s_i$. Verification is very similar to the verification of Schnorr signatures. As each signer in this simple two-round scheme is almost identical to the prover algorithm of the Schnorr identification scheme, one may hope that this scheme is secure. However, early works already noted that it is not [5].

While there are concrete attacks against the scheme, for our purposes it is more important to understand where the security proof fails. The proof fails when we try to simulate honest signer without knowing its secret key sk_1. Following Schnorr signatures and identification, this would be done by sampling $R_1 := \mathsf{F}(s_1) - c_1\mathsf{pk}_1$ for random c_1 an s_1, and then programming the random oracle accordingly at position R. The problem in the multi-signature setting is that we first have to output R_1, and then the adversary can output the remaining R_i, such that he has full control over the aggregate R. Thus, the random oracle may already be defined. Previous works [5,30,8] solve this issue by introducing an additional

round, in which all signers commit to their R_i using a random oracle. This allows us to extract all R_i from these commitments in the reduction, and therefore R has enough entropy to program the random oracle.

A second problem that we encounter in the above approach is the extraction of a solution from the forgery. Namely, to extract a discrete logarithm of pk_1, we need to rely on rewinding. Some of the well-known schemes [30,8] even use rewinding multiple times. This leads to security bounds with essentially no useful quantitative guarantee for concrete security.

Towards A Scheme without Rewinding. To avoid rewinding, our first idea is to rely on a different homomorphism F. Namely, we borrow techniques from lossy identification [26,1,27] and use $\mathsf{F} : x \mapsto (g^x, h^x)$ for a second generator $h \in \mathbb{G}$. We can then give a non-rewinding security proof for the three-round schemes in [5,30,8]. Concretely, we first switch pk_1 from the range of F to a random element in $\mathbb{G}^2$, using the DDH assumption. Then, we can argue that a forgery is hard to compute using a statistical argument. We note that this idea is (implicitly) already present in [5,16]. As we will see, combining it with techniques to avoid the extra round is challenging.

Towards Two-Round Schemes. To go from a three-round scheme as above to a two-round scheme, our goal is to avoid the first round. Recall that this round was needed to simulate R_1 using random oracle programming. Our idea to tackle the simulation problem is a bit different. Namely, going back to the (insecure) two-round scheme, our goal is to send R_1 *after* we learn c_1. If we manage to do that, we can simulate by setting it as $R_1 := \mathsf{F}(s_1) - c_1\mathsf{pk}_1$ for random s_1. Of course, just sending R_1 after learning c_1 should only be possible for the reduction. Following Damgård [12], this high-level strategy can be implemented using a trapdoor commitment scheme Com, and sending $\mathsf{com}_1 = \mathsf{Com}(\mathsf{ck}, R_1)$ as the first message. The challenges c_i are then derived from an aggregated commitment com using the random oracle. Later, the reduction can open this commitment to $\mathsf{F}(s_1) - c_1\mathsf{pk}_1$ using the trapdoor for commitment key ck. To support aggregation, the commitment scheme should have homomorphic properties. Note that this approach has been used in the lattice setting in a recent work [13]. However, implementing such a commitment scheme for (pairs of) group elements is highly non-trivial, as we will see. Also, as already pointed out in [13], it is hard to make this two-round approach work while avoiding rewinding at the same time. The reason is that a trapdoor commitment scheme can not be statistically binding. But if we want to make use of the statistical argument from lossy identification discussed above, we need that R is fixed before the c_i are sampled, which requires statistical binding. With a computationally binding commitment scheme, we end up in a rewinding reduction (to binding) again. Our first technical main contribution is to overcome this issue.

Chopstick One: Our Scheme Without Rewinding. Our idea to overcome the above problem is to demand a dual-mode property from the commitment scheme Com. Namely, there should be an indistinguishable second way to set up the

commitment key ck, such that for such a key the scheme is statistically binding. This does not solve the problem yet, because we require ck to be in trapdoor mode for simulation, and in binding mode for the final forgery. The solution is to sample ck in a message-dependent way using another random oracle, which is (for other reasons) already done in earlier works [14,13]. In this way, we can embed a binding commitment key in some randomly guessed random oracle queries, and a trapdoor key in others. Note that this requires a tight multi-key indistinguishability of the commitment scheme. Assuming we have such a commitment scheme, we end up with our first construction, which is presented formally in Section 3.2. Of course, this strategy still has a security loss linear in the number of signing queries due to the guessing argument, but it avoids rewinding, leading to an acceptable security bound. In addition, we can implement the approach in a way that supports key aggregation.

Chopstick Two: Our Fully Tight Scheme. The security loss in our first scheme results from partitioning random oracle queries into two classes, namely queries returning binding keys, and queries returning trapdoor keys. To do such a partitioning in a tight way, we may try to use a Katz-Wang random bit approach [17]. This simple approach can be used in standard digital signatures. However, it turns out that it does not work for our case. To see this, recall that following this approach, we would compute two message-dependent commitment keys

$$\mathsf{ck}_0 := \mathsf{H}(0, \mathsf{m}), \quad \mathsf{ck}_1 := \mathsf{H}(1, \mathsf{m}).$$

Then, for each message, we would embed a binding key in one branch, and a trapdoor key in the other branch, e.g. ck_0 binding and ck_1 with trapdoor. In the signing protocol, we would abort one of the branches pseudorandomly based on the message. Then we could use the trapdoor branch in the signing, and hope that the forgery uses the binding branch. However, this strategy crucially relies on the fact that the aborting happens in a way that is pseudorandom to the adversary. Otherwise the adversary could always choose the trapdoor branch for his forgery. While we can implement this in a signature scheme, in our multi-signature scheme this fails, because all signers must use the *same commitment key* to make aggregation possible. At the same time, the aborted branch must depend on secret data of the simulated signer to remain pseudorandom.

To solve this problem, we observe that the above approach uses a pseudorandom "branch selection" and aborts the other branch. Our solution can be phrased as a pseudorandom "branch-to-key matching". Namely, we give each signer two public keys $(\mathsf{pk}_{i,0}, \mathsf{pk}_{i,1})$. The signing protocol is run in two instances in parallel. One instance uses ck_0, and one uses ck_1 as above. More precisely, we commit to R_0 via ck_0 and to R_1 via ck_1. Then we aggregate and determine the challenges $c_{i,0}$ and $c_{i,1}$. However, before sending the response $s_i = (s_{i,0}, s_{i,1})$, *each signer separately* determines which key to use in which instance, i.e. it computes

$$s_{i,0} = c_{i,0} \cdot x_{i,b_i} + r_{i,0}, \quad s_{i,1} = c_{i,1} \cdot x_{i,1-b_i} + r_{i,1},$$

where b_i is a pseudorandom bit that each signer i computes *independently*, and that will be included in the final signature to make verification possible. This

decouples the public key that is used from the commitment key that is used. Now we are ready to discuss the implication of this change. Namely, our reduction chooses $\mathsf{pk}_{1,0}$ honestly and $\mathsf{pk}_{1,1}$ as a lossy key, i.e. random instead of in the range of F. Then, in each signing interaction, the reduction can match the honest public key with the binding commitment key and the lossy public key with the trapdoor commitment key by setting b_1 accordingly. In this way, we can simulate one branch using the actual secret key, and the other branch using the commitment trapdoor. For the forgery, we hope that the matching is the other way around, such that binding commitment key and lossy public key match, which makes the statistical argument from lossy identification possible. Overall, this approach leads to our fully tight scheme, presented in Section 3.3.

The Challenge of Instantiating the Commitment. One may observe that we shifted a lot of the challenges that we encountered into properties of the underlying commitment scheme. This naturally raises the question if such a commitment scheme can be found. In fact, constructing this commitment scheme can be understood as our second technical main contribution.

Let us first explain why it is non-trivial to construct such a scheme. The main barrier results from the algebraic structure that we demand. Namely, we need to commit to group elements[6] $R \in \mathbb{G}$. A naive idea would be to use any trapdoor commitment scheme, e.g. Pedersen commitments, by first encoding R in the appropriate message space. However, this would destroy all homomorphic properties that we need, and we should not forget that we need a dual-mode property. This brings us to Groth-Sahai commitments [20], which can commit to group elements. Indeed, these commitments are homomorphic, and have (indistinguishable from) random keys, such that we can sample them using a random oracle. They are also dual-mode based on DDH, which allows us to use the random self-reducibility of DDH to show tight multi-key indistinguishability. However, the trapdoor property turns out to be the main challenge. To see why this is problematic, note that the opening information of these commitments typically contains elements from $\mathbb{Z}_p$ that are somehow used as exponents. There are exceptions to this rule, like [18], but they use pairings and the DLIN assumption, which we aim to avoid. This means that the trapdoor should allow us to sample exponents, given a group element R to which we want to open the commitment. This naturally corresponds to having a trapdoor for the discrete logarithm problem, which we do not have.

Our Solution: Weakly Equivocable Commitments. Our starting point is the commitment scheme for group elements given in [20]. Namely, commitment keys correspond to matrices $\mathbf{A} = (A_{i,j})_{i,j} \in \mathbb{G}^{2\times 2}$, and to commit to a message $R = g^r \in \mathbb{G}$ with randomness $(\alpha, \beta) \in \mathbb{Z}_p$, one computes

$$\mathsf{com} := (C_0, C_1)^t := \left(A_{1,1}^{\alpha} \cdot A_{1,2}^{\beta}, R \cdot A_{2,1}^{\alpha} \cdot A_{2,2}^{\beta}\right)^t .$$

[6] In the actual construction, we need to commit to pairs of group elements, but we consider the simpler setting of one group element in this overview.

That is, setting $\mathbf{E} = (E_{i,j})_{i,j} \in \mathbb{Z}_p$ such that $g^{E_{i,j}} = A_{i,j}$, we can write the discrete logarithm of com as $(0,r)^t + \mathbf{E} \cdot (\alpha,\beta)^t$. In binding mode, matrix $\mathbf{E}$ is a matrix of rank 1, while $\mathbf{E}$ has full rank in hiding mode. It is easy to see that this commitment scheme to group elements is homomorphic. However, we stress that there is no simple solution to implement a trapdoor for equivocation. To see this, note that if we want to open a commitment com to a message $R' \in \mathbb{G}$, we need to output a suitable tuple (α,β). If we knew the discrete logarithm of com, then we still would need to know the discrete logarithm of R' to find such a tuple. The key insight of our trapdoor construction is that we do not need to be able to open com to any message R'. Instead, it will be sufficient if we can open it to messages of the form $R' = g^s \cdot \mathsf{pk}^c$, where we do not know c when we fix the commitment com, but *we know* pk *when setting up* $\mathbf{A}$. To explain why this helps, assume we want to find a valid opening (α,β) in this case. Then we need to satisfy

$$\mathsf{com} = \begin{pmatrix} C_0 \\ C_1 \end{pmatrix} = \begin{pmatrix} 0 \\ g^s \mathsf{pk}^c \end{pmatrix} \cdot g^{\mathbf{E}\cdot(\alpha,\beta)^t}.$$

It seems like we did not make progress, because even if we know the discrete logarithms of C_0, C_1, the term pk^c is not known in the exponent. Now, our key idea to solve this is to write and generate $\mathbf{A}$ with respect to basis pk in the second row. Namely, we generate $\mathbf{A}$ as

$$\mathbf{A} = \begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} := \begin{pmatrix} g^{d_{1,1}} & g^{d_{1,2}} \\ \mathsf{pk}^{d_{2,1}} & \mathsf{pk}^{d_{2,2}} \end{pmatrix}.$$

In this way, the equation that we need to satisfy becomes

$$\begin{pmatrix} C_0 \\ C_1 \end{pmatrix} = \begin{pmatrix} g^{d_{1,1}\alpha + d_{1,2}\beta} \\ g^s \mathsf{pk}^{c + d_{2,1}\alpha + d_{2,2}\beta} \end{pmatrix}.$$

Next, we get rid of the term g^s by shifting C_1 accordingly. Namely, recall that we can sample s at random long before we learn c. Setting $C_0 = g^\tau$ and $C_1 = g^s \mathsf{pk}^\rho$ for random τ, ρ, we obtain the equation

$$\begin{pmatrix} g^\tau \\ \mathsf{pk}^\rho \end{pmatrix} = \begin{pmatrix} g^{d_{1,1}\alpha + d_{1,2}\beta} \\ \mathsf{pk}^{c + d_{2,1}\alpha + d_{2,2}\beta} \end{pmatrix}.$$

Given the trapdoor $\mathbf{D} = (d_{i,j})_{i,j}$, this can easily be solved for (α,β) by solving $(\tau, \rho - c)^t = \mathbf{D} \cdot (\alpha,\beta)^t$. We are confident that such a weak and structured equivocation property can be used in other applications as well, and formally define this type of commitment scheme in Section 3.1.

2 Preliminaries

We denote the security parameter by $\lambda \in \mathbb{N}$, and all algorithms get 1^λ implicitly as input. We write $x \xleftarrow{\$} S$ if x is sampled uniformly at random from a finite set S, and we write $x \leftarrow \mathcal{D}$ if x is sampled according to a distribution $\mathcal{D}$. We

write $y \leftarrow \mathcal{A}(x)$, if y is output from (probabilistic) algorithm $\mathcal{A}$ on input x with uniform coins. To make the coins explicit, we use the notation $y = \mathcal{A}(x; \rho)$. The notation $y \in \mathcal{A}(x)$ indicates that y is a possible output of $\mathcal{A}(x)$. We use standard asymptotic notation, and the notions of negligible functions, and PPT algorithms. If $\mathbf{G}$ is a security game, we write $\mathbf{G} \Rightarrow b$ to state that $\mathbf{G}$ outputs b. In all our games, numerical variables are implicitly initialized with 0, and lists and sets are initialized with $\emptyset$. We define $[K] := \{1, \ldots, K\}$, and denote the Bernoulli distribution with parameter $\gamma \in [0, 1]$ by $\mathcal{B}_\gamma$.

Multi-Signatures. We introduce syntax and security for multi-signatures, following the established security notions in the plain public key model [5]. We will assume that there is an canonical ordering of given multi-sets, e.g. lexicographically, that allows us to uniquely encode multi-sets $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$. For this encoding, we write $\langle \mathcal{P} \rangle$ throughout the paper. Further, for simplicity of notation, we assume that the honest public key in our security definition is the entry pk_1 in this multi-set.

Alg $\mathsf{MS.Exec}(\mathcal{P}, \mathcal{S}, \mathsf{m})$
01 **let** $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\},\ \mathcal{S} = \{\mathsf{sk}_1, \ldots, \mathsf{sk}_N\}$
02 **for** $i \in [N] : (\mathsf{pm}_{1,i}, St_{1,i}) \leftarrow \mathsf{Sig}_0(\mathcal{P}, \mathsf{sk}, \mathsf{m})$
03 $\mathcal{M}_1 := (\mathsf{pm}_{1,1}, \ldots, \mathsf{pm}_{1,N})$
04 **for** $i \in [N] : (\mathsf{pm}_{2,i}, St_{2,i}) \leftarrow \mathsf{Sig}_1(St_{1,i}, \mathcal{M}_1)$
05 $\mathcal{M}_2 := (\mathsf{pm}_{2,1}, \ldots, \mathsf{pm}_{2,N})$
06 **for** $i \in [N] : \sigma_i \leftarrow \mathsf{Sig}_2(St_{2,i}, \mathcal{M}_2)$
07 **if** $\exists i \neq j \in [N]$ s.t. $\sigma_i \neq \sigma_j$: **return** $\perp$
08 **return** $\sigma := \sigma_1$

Fig. 1. The algorithm $\mathsf{MS.Exec}$ for a (two-round) multi-signature scheme $\mathsf{MS} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$, representing an honest execution of the signing protocol Sig.

Definition 1 (Multi-Signature Scheme). *A (two-round) multi-signature scheme is a tuple of PPT algorithms* $\mathsf{MS} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ *with the following syntax:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{par}$ *takes as input the security parameter* 1^λ *and outputs global system parameters* par*. We assume that* par *implicitly defines sets of public keys, secret keys, messages and signatures, respectively. All algorithms related to* SIG *take at least implicitly* par *as input.*
- $\mathsf{Gen}(\mathsf{par}) \to (\mathsf{pk}, \mathsf{sk})$ *takes as input system parameters* par*, and outputs a public key* pk *and a secret key* sk*.*
- $\mathsf{Sig} = (\mathsf{Sig}_0, \mathsf{Sig}_1, \mathsf{Sig}_2)$ *is split into three algorithms:*
  - $\mathsf{Sig}_0(\mathcal{P}, \mathsf{sk}, \mathsf{m}) \to (\mathsf{pm}_1, St_1)$ *takes as input a multi-set* $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$ *of public keys, a secret key* sk*, and a message* m*, and outputs a protocol message* pm_1 *and a state* St_1*.*

  - $\mathsf{Sig}_1(St_1, \mathcal{M}_1) \to (\mathsf{pm}_2, St_2)$ *takes as input a state* St_1 *and a tuple* $\mathcal{M}_1 = (\mathsf{pm}_{1,1}, \ldots, \mathsf{pm}_{1,N})$ *of protocol messages, and outputs a protocol message* pm_2 *and a state* St_2*.*
  - $\mathsf{Sig}_2(St_2, \mathcal{M}_2) \to \sigma_i$ *takes as input a state* St_2 *and a tuple* $\mathcal{M}_2 = (\mathsf{pm}_{2,1}, \ldots, \mathsf{pm}_{2,N})$ *of protocol messages, and outputs a signature* σ*.*
- $\mathsf{Ver}(\mathcal{P}, \mathsf{m}, \sigma) \to b$ *is deterministic, takes as input a multi-set* $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$ *of public keys, a message* m*, and a signature* σ*, and outputs a bit* $b \in \{0,1\}$*.*

We require that MS *is complete, i.e. for all* $\mathsf{par} \in \mathsf{Setup}(1^\lambda)$*, all* $N = \mathsf{poly}(\lambda)$*, all* $(\mathsf{pk}_j, \mathsf{sk}_j) \in \mathsf{Gen}(\mathsf{par})$ *for* $j \in [N]$*, and all messages* m*, we have*

$$\Pr\left[\mathsf{Ver}(\mathcal{P}, \mathsf{m}, \sigma) = 1 \;\middle|\; \begin{array}{l} \mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}, \mathcal{S} = \{\mathsf{sk}_1, \ldots, \mathsf{sk}_N\}, \\ \sigma \leftarrow \mathsf{MS.Exec}(\mathcal{P}, \mathcal{S}, \mathsf{m}) \end{array}\right] = 1,$$

where algorithm MS.Exec *is defined in Figure 1.*

Definition 2 (Key Aggregation). *A multi-signature scheme* $\mathsf{MS} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ *is said to support key aggregation, if the algorithm* Ver *can be split into two deterministic polynomial time algorithms* $\mathsf{Agg}, \mathsf{VerAgg}$ *with the following syntax:*

- $\mathsf{Agg}(\mathcal{P}) \to \tilde{\mathsf{pk}}$ *takes as input a multi-set* $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$ *of public keys and outputs an aggregated key* $\tilde{\mathsf{pk}}$*.*
- $\mathsf{VerAgg}(\tilde{\mathsf{pk}}, \mathsf{m}, \sigma) \to b$ *is deterministic, takes as input an aggregated key* $\tilde{\mathsf{pk}}$*, a message* m*, and a signature* σ*, and outputs a bit* $b \in \{0,1\}$*.*

Precisely, algorithm $\mathsf{Ver}(\mathcal{P}, \mathsf{m}, \sigma)$ *can be written as* $\mathsf{VerAgg}(\mathsf{Agg}(\mathcal{P}), \mathsf{m}, \sigma)$*.*

Definition 3 (MS-EUF-CMA Security). *Let* $\mathsf{MS} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ *be a multi-signature scheme and consider the game* **MS-EUF-CMA** *defined in Figure 2. We say that* MS *is* MS-EUF-CMA *secure, if for all PPT adversaries* $\mathcal{A}$*, the following advantage is negligible:*

$$\mathsf{Adv}^{\mathsf{MS\text{-}EUF\text{-}CMA}}_{\mathcal{A},\mathsf{MS}}(\lambda) := \Pr\left[\mathbf{MS\text{-}EUF\text{-}CMA}^{\mathcal{A}}_{\mathsf{MS}}(\lambda) \Rightarrow 1\right].$$

Linear Function Families. To present our constructions in a modular way, we make use of the abstraction of linear function families. Our definition is close to previous definitions [23,25,10]. As it is not needed for our instantiations, we restrict our setting to vector spaces instead of pseudo modules.

Definition 4 (Linear Function Family). *A linear function family (LFF) is a tuple of PPT algorithms* $\mathsf{LF} = (\mathsf{Gen}, \mathsf{F})$ *with the following syntax:*

- $\mathsf{Gen}(1^\lambda) \to \mathsf{par}$ *takes as input the security parameter* 1^λ *and outputs parameters* par*. We assume that* par *implicitly defines the following sets:*
  - *A set of scalars* $\mathcal{S}_{\mathsf{par}}$*, which forms a field.*
  - *A domain* $\mathcal{D}_{\mathsf{par}}$*, which forms a vector space over* $\mathcal{S}_{\mathsf{par}}$*.*
  - *A range* $\mathcal{R}_{\mathsf{par}}$*, which forms vector space over* $\mathcal{S}_{\mathsf{par}}$*.*

  We omit the subscript par *if it is clear from the context, and naturally denote the operations of these fields and vector spaces by* $+$ *and* $\cdot$*.*

Game MS-EUF-CMA$^{\mathcal{A}}_{\mathsf{MS}}(\lambda)$

$\mathsf{par} \leftarrow \mathsf{Setup}(1^\lambda)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{par})$
$\textsc{Sig} := (\textsc{Sig}_0, \textsc{Sig}_1, \textsc{Sig}_2)$
$(\mathcal{P}^*, \mathsf{m}^*, \sigma^*) \leftarrow \mathcal{A}^{\textsc{Sig}}(\mathsf{par}, \mathsf{pk})$
if $\mathsf{pk} \notin \mathcal{P}^* \vee (\mathcal{P}^*, \mathsf{m}^*) \in \mathcal{L}$:
  return 0
return $\mathsf{Ver}(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$

Oracle $\textsc{Sig}_0(\mathcal{P}, \mathsf{m})$

let $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$
if $\mathsf{pk}_1 \neq \mathsf{pk}$: **return** $\perp$
$\mathcal{L} := \mathcal{L} \cup \{(\mathcal{P}, \mathsf{m})\}$
$sid := sid + 1,\ ctr[sid] := 1$
$(\mathsf{pm}_1, St_1) \leftarrow \mathsf{Sig}_0(\mathcal{P}, \mathsf{sk}, \mathsf{m})$
$(\mathsf{pm}_1[sid], St_1[sid]) := (\mathsf{pm}_1, St_1)$
return $(\mathsf{pm}_1[sid], sid)$

Oracle $\textsc{Sig}_1(sid, \mathcal{M}_1)$

if $ctr[sid] \neq 1$: **return** $\perp$
let $\mathcal{M}_1 = (\mathsf{pm}_{1,1}, \ldots, \mathsf{pm}_{1,N})$
if $\mathsf{pm}_1[sid] \neq \mathsf{pm}_{1,1}$: **return** $\perp$
$ctr[sid] := ctr[sid] + 1$
$(\mathsf{pm}_2, St_2) \leftarrow \mathsf{Sig}_1(St_1[sid], \mathcal{M}_1)$
$(\mathsf{pm}_2[sid], St_2[sid]) := (\mathsf{pm}_2, St_2)$
return $\mathsf{pm}_2[sid]$

Oracle $\textsc{Sig}_2(sid, \mathcal{M}_2)$

if $ctr[sid] \neq 2$: **return** $\perp$
let $\mathcal{M}_2 = (\mathsf{pm}_{2,1}, \ldots, \mathsf{pm}_{2,N})$
if $\mathsf{pm}_2[sid] \neq \mathsf{pm}_{2,1}$: **return** $\perp$
$ctr[sid] := ctr[sid] + 1$
$\sigma \leftarrow \mathsf{Sig}_2(St_2[sid], \mathcal{M}_2)$
return σ

Fig. 2. The game **MS-EUF-CMA** for a (two-round) multi-signature scheme MS and an adversary $\mathcal{A}$. For simplicity of exposition, we assume that the canonical ordering of multi-sets is chosen such that pk is always at the first position if it is included.

– $\mathsf{F}(\mathsf{par}, x) \to X$ *is deterministic, takes as input parameters* par*, an element* $x \in \mathcal{D}$*, and outputs an element* $X \in \mathcal{R}$*. For all parameters* par*,* $\mathsf{F}(\mathsf{par}, \cdot)$ *realizes a homomorphism, i.e.*

$$\forall s \in \mathcal{S}, x, y \in \mathcal{D} : \ \mathsf{F}(\mathsf{par}, s \cdot x + y) = s \cdot \mathsf{F}(\mathsf{par}, x) + \mathsf{F}(\mathsf{par}, y).$$

We omit the input par *if it is clear from the context.*

We formalize necessary conditions under which a linear function family can be used to construct so called lossy identification [1]. Our constructions will rely on such linear function families. We also give a similar definition that captures a similar property in the context of key aggregation.

Definition 5 (Lossiness Admitting LFF). *We say that a linear function family* $\mathsf{LF} = (\mathsf{Gen}, \mathsf{F})$ *is* ε_{l}*-lossiness admitting, if the following properties hold:*

– ***Key Indistinguishability.*** *For any PPT algorithm* $\mathcal{A}$*, the following advantage is negligible:*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{keydist}}_{\mathcal{A},\mathsf{LF}}(\lambda) := |\Pr\left[\mathcal{A}(\mathsf{par}, X) = 1 \middle| \mathsf{par} \leftarrow \mathsf{Gen}(1^\lambda),\ x \xleftarrow{\$} \mathcal{D},\ X := \mathsf{F}(x)\right] \\ - \Pr\left[\mathcal{A}(\mathsf{par}, X) = 1 \middle| \mathsf{par} \leftarrow \mathsf{Gen}(1^\lambda),\ X \xleftarrow{\$} \mathcal{R}\right]|.\end{aligned}$$

– ***Lossy Soundness.*** *For any unbounded algorithm* $\mathcal{A}$*, the following probability is at most* ε_{l}*:*

$$\Pr\left[\mathsf{F}(s) - c \cdot X = R \;\middle|\; \begin{array}{l}\mathsf{par} \leftarrow \mathsf{Gen}(1^\lambda),\ X \xleftarrow{\$} \mathcal{R}, \\ (St, R) \leftarrow \mathcal{A}(\mathsf{par}, X), \\ c \xleftarrow{\$} \mathcal{S},\ s \leftarrow \mathcal{A}(St, c)\end{array}\right].$$

Definition 6 (Aggregation Lossy Soundness). *We say that a linear function family* $\mathsf{LF} = (\mathsf{Gen}, \mathsf{F})$ *satisfies* $\varepsilon_{\mathsf{al}}$*-aggregation lossy soundness, if for any unbounded algorithm* $\mathcal{A}$*, the following probability is at most* $\varepsilon_{\mathsf{al}}$*:*

$$\Pr\left[\mathsf{F}(s) - c \cdot \sum_{i=1}^{N} a_i X_i = R \;\middle|\; \begin{array}{l} \mathsf{par} \leftarrow \mathsf{Gen}(1^\lambda),\ X_1 \xleftarrow{\$} \mathcal{R}, \\ (St, (X_2, a_2), \ldots, (X_N, a_N)) \leftarrow \mathcal{A}(\mathsf{par}, X_1), \\ a_1 \xleftarrow{\$} \mathcal{S},\ (St', R) \leftarrow \mathcal{A}(St, a_1), \\ c \xleftarrow{\$} \mathcal{S},\ s \leftarrow \mathcal{A}(St', c) \end{array}\right].$$

Assumptions. We recall the computational assumptions that we need.

Definition 7 (DDH Assumption). *Let* GGen *be an algorithm that on input* 1^λ *outputs the description of a prime order group* $\mathbb{G}$ *of order* p *with generator* g*. We say that the* DDH *assumption holds relative to* GGen*, if for all PPT algorithms* $\mathcal{A}$*, the following advantage is negligible:*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{A},\mathsf{GGen}}(\lambda) := |\Pr\left[\mathcal{A}(\mathbb{G}, p, g, h, g^a, h^a) = 1 \;\middle|\; \begin{array}{l}(\mathbb{G}, g, p) \leftarrow \mathsf{GGen}(1^\lambda), \\ h \xleftarrow{\$} \mathbb{G}, a \xleftarrow{\$} \mathbb{Z}_p\end{array}\right] \\ - \Pr\left[\mathcal{A}(\mathbb{G}, p, g, h, g^a, g^b) = 1 \;\middle|\; \begin{array}{l}(\mathbb{G}, g, p) \leftarrow \mathsf{GGen}(1^\lambda), \\ h \xleftarrow{\$} \mathbb{G}, a, b \xleftarrow{\$} \mathbb{Z}_p\end{array}\right]|.\end{aligned}$$

In the following, we define an equivalent variant of the DDH assumption, uDDH3. uDDH3 is the 2-fold $\mathcal{U}_{3,1}$-Matrix-DDH (MDDH) assumption (with terminology in [15]). By its random self-reducibility [15, Lemma 1], the 2-fold $\mathcal{U}_{3,1}$-Matrix-DDH (MDDH) assumption (namely, the uDDH3 assumption) is tightly equivalent to the $\mathcal{U}_{3,1}$-MDDH assumption. By Lemma 1 in [28], $\mathcal{U}_{3,1}$-MDDH is tightly equivalent to $\mathcal{U}_1$-MDDH that is the DDH assumption. Hence, the DDH and uDDH3 assumptions are tightly equivalent.

Definition 8 (uDDH3 Assumption). *Let* GGen *be an algorithm that on input* 1^λ *outputs the description of a prime order group* $\mathbb{G}$ *of order* p *with generator* g*. We say that the* uDDH3 *assumption holds relative to* GGen*, if for all PPT algorithms* $\mathcal{A}$*, the following advantage is negligible:*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{uDDH3}}_{\mathcal{A},\mathsf{GGen}}(\lambda) := |\Pr\left[\mathcal{A}(\mathbb{G}, p, g, (h_{i,j})_{i,j\in[3]}) = 1 \;\middle|\; \begin{array}{l}(\mathbb{G}, g, p) \leftarrow \mathsf{GGen}(1^\lambda), \\ a, b \xleftarrow{\$} \mathbb{Z}_p, \\ h_{1,1}, h_{2,1}, h_{3,1} \xleftarrow{\$} \mathbb{G} \\ h_{1,2} := h_{1,1}^a, h_{1,3} := h_{1,1}^b \\ h_{2,2} := h_{2,1}^a, h_{2,3} := h_{2,1}^b \\ h_{3,2} := h_{3,1}^a, h_{3,3} := h_{3,1}^b\end{array}\right] \\ - \Pr\left[\mathcal{A}(\mathbb{G}, p, g, (h_{i,j})_{i,j\in[3]}) = 1 \;\middle|\; \begin{array}{l}(\mathbb{G}, g, p) \leftarrow \mathsf{GGen}(1^\lambda), \\ \forall (i,j) \in [3] \times [3] : h_{i,j} \xleftarrow{\$} \mathbb{G}\end{array}\right]|.\end{aligned}$$

3 Constructions

In this section, we present our construction of two-round multi-signatures. First, we give a definition of a special commitment scheme that will be used in both constructions. Then, we present the constructions in an abstract way. For the instantiation, we refer to Section 4.

3.1 Preparation: Special Commitments

In this section we define a special kind of commitment scheme. We will make use of such a scheme in our constructions of multi-signatures. Before we give the definition, we explain the desired properties at a high level. First of all, we want to be able to commit to elements $R \in \mathcal{R}$ in the range of a given linear function family. Second, we need the commitment scheme to be homomorphic in both messages and randomness, allowing us to aggregate commitments during the signing protocol. Third, we need a certain dual mode property, ensuring that we can set up keys either in a perfectly hiding or in a perfectly binding mode. This will allow us to make the commitment key for the forgery binding, while associating a equivocation trapdoor to the keys used to answer signing queries. We emphasize that we do not need a full-fledged equivocation feature. This is because we already know parts of the structure of messages to which we want to open the commitment. Looking ahead, this is the reason we can instantiate the commitment in the DDH setting.

Game Q-$\mathbf{KEYDIST}^{\mathcal{A}}_{0,\mathsf{CMT}}(\lambda)$	**Game** Q-$\mathbf{KEYDIST}^{\mathcal{A}}_{1,\mathsf{CMT}}(\lambda)$
01 $\mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda),\ x \xleftarrow{\$} \mathcal{D}$	06 $\mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda),\ x \xleftarrow{\$} \mathcal{D}$
02 **if** $(\mathsf{par}, x) \notin \mathsf{Good}$: **return** 0	07 **if** $(\mathsf{par}, x) \notin \mathsf{Good}$: **return** 0
03 **for** $i \in [Q]$: $\mathsf{ck}_i \leftarrow \mathsf{BGen}(\mathsf{par})$	08 **for** $i \in [Q]$: $\mathsf{ck}_i \xleftarrow{\$} \mathcal{K}_{\mathsf{par}}$
04 $\beta \leftarrow \mathcal{A}(\mathsf{par}, x, (\mathsf{ck}_i)_{i\in[Q]})$	09 $\beta \leftarrow \mathcal{A}(\mathsf{par}, x, (\mathsf{ck}_i)_{i\in[Q]})$
05 **return** β	10 **return** β

Fig. 3. The games $\mathbf{KEYDIST}_0, \mathbf{KEYDIST}_1$ for a special commitment scheme CMT and an adversary $\mathcal{A}$.

Definition 9 (Special Commitment Scheme). *Let* $\mathsf{LF} = (\mathsf{LF.Gen}, \mathsf{F})$ *be a linear function family and* $\mathcal{G} = \{\mathcal{G}_{\mathsf{par}}\}, \mathcal{H} = \{\mathcal{H}_{\mathsf{par}}\}$ *be families of subsets of abelian groups with efficiently computable group operations* $\oplus$ *and* $\otimes$*, respectively. Let* $\mathcal{K} = \{\mathcal{K}_{\mathsf{par}}\}$ *be a family of sets. An* $(\varepsilon_{\mathsf{b}}, \varepsilon_{\mathsf{g}}, \varepsilon_{\mathsf{t}})$*-special commitment scheme for* LF *with key space* $\mathcal{K}$*, randomness space* $\mathcal{G}$ *and commitment space* $\mathcal{H}$ *is a tuple of PPT algorithms* $\mathsf{CMT} = (\mathsf{BGen}, \mathsf{TGen}, \mathsf{Com}, \mathsf{TCom}, \mathsf{TCol})$ *with the following syntax:*

- $\mathsf{BGen}(\mathsf{par}) \to \mathsf{ck}$ *takes as input parameters* par*, and outputs a key* $\mathsf{ck} \in \mathcal{K}_{\mathsf{par}}$*.*
- $\mathsf{TGen}(\mathsf{par}, X) \to (\mathsf{ck}, \mathsf{td})$ *takes as input parameters* par*, and an element* $X \in \mathcal{R}$*, and outputs a key* $\mathsf{ck} \in \mathcal{K}_{\mathsf{par}}$ *and a trapdoor* td*.*
- $\mathsf{Com}(\mathsf{ck}, R; \varphi) \to \mathsf{com}$ *takes as input a key* ck*, an element* $R \in \mathcal{R}$*, and a randomness* $\varphi \in \mathcal{G}_{\mathsf{par}}$*, and outputs a commitment* $\mathsf{com} \in \mathcal{H}_{\mathsf{par}}$*.*
- $\mathsf{TCom}(\mathsf{ck}, \mathsf{td}) \to (\mathsf{com}, St)$ *takes as input a key* ck *and a trapdoor* td*, and outputs a commitment* $\mathsf{com} \in \mathcal{H}_{\mathsf{par}}$ *and a state* St*.*
- $\mathsf{TCol}(St, c) \to (\varphi, R, s)$ *takes as input a state* St*, and an element* $c \in \mathcal{S}$*, and outputs randomness* $\varphi \in \mathcal{G}_{\mathsf{par}}$*, and elements* $R \in \mathcal{R}, s \in \mathcal{D}$*.*

We omit the subscript par *if it is clear from the context.*

Further, the algorithms are required to satisfy the following properties:

- ***Homomorphism.*** *For all* $\mathsf{par} \in \mathsf{LF.Gen}(1^\lambda), \mathsf{ck} \in \mathcal{K}_{\mathsf{par}}, R_0, R_1 \in \mathcal{R}$ *and* $\varphi_0, \varphi_1 \in \mathcal{G}$*, the following holds:*

$$\mathsf{Com}(\mathsf{ck}, R_0; \varphi_0) \otimes \mathsf{Com}(\mathsf{ck}, R_1; \varphi_1) = \mathsf{Com}(\mathsf{ck}, R_0 + R_1; \varphi_0 \oplus \varphi_1).$$

- ***Good Parameters.*** *There is a set* Good*, such that membership to* Good *can be decided in polynomial time, and*

$$\Pr\left[(\mathsf{par}, x) \notin \mathsf{Good} \mid \mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda),\ x \xleftarrow{\$} \mathcal{D}\right] \leq \varepsilon_{\mathsf{g}},$$

- ***Uniform Keys.*** *For all* $(\mathsf{par}, x) \in \mathsf{Good}$*, the following distributions are identical:*

$$\{(\mathsf{par}, x, \mathsf{ck}) \mid \mathsf{ck} \xleftarrow{\$} \mathcal{K}_{\mathsf{par}}\} \text{ and } \{(\mathsf{par}, x, \mathsf{ck}) \mid (\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{TGen}(\mathsf{par}, X)\}.$$

- ***Special Trapdoor Property.*** *For all* $(\mathsf{par}, x) \in \mathsf{Good}$*, and all* $c \xleftarrow{\$} \mathcal{S}$*, the following distributions* $\mathcal{T}_0$ *and* $\mathcal{T}_1$ *have statistical distance at most* ε_{t}*:*

$$\mathcal{T}_0 := \left\{ (\mathsf{par}, \mathsf{ck}, \mathsf{td}, x, c, \mathsf{com}, \mathsf{tr}) \;\middle|\; \begin{array}{l} (\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{TGen}(\mathsf{par}, \mathsf{F}(x)) \\ (\mathsf{com}, St) \leftarrow \mathsf{TCom}(\mathsf{ck}, \mathsf{td}), \\ \mathsf{tr} \leftarrow \mathsf{TCol}(St, c) \end{array} \right\}$$

$$\mathcal{T}_1 := \left\{ (\mathsf{par}, \mathsf{ck}, \mathsf{td}, x, c, \mathsf{com}, \mathsf{tr}) \;\middle|\; \begin{array}{l} (\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{TGen}(\mathsf{par}, \mathsf{F}(x)) \\ r \xleftarrow{\$} \mathcal{D},\ R := \mathsf{F}(r),\ \varphi \xleftarrow{\$} \mathcal{G}, \\ \mathsf{com} := \mathsf{Com}(\mathsf{ck}, R; \varphi), \\ s := c \cdot x + r,\ \mathsf{tr} := (\varphi, R, s) \end{array} \right\}$$

- ***Multi-Key Indistinguishability.*** *For every* $Q = \mathsf{poly}(\lambda)$ *and any PPT algorithm* $\mathcal{A}$*, the following advantage is negligible:*

$$\begin{aligned} \mathsf{Adv}^{Q\text{-}\mathsf{keydist}}_{\mathcal{A},\mathsf{CMT}}(\lambda) := |\Pr\left[Q\text{-}\mathbf{KEYDIST}^{\mathcal{A}}_{0,\mathsf{CMT}}(\lambda) \Rightarrow 1\right] \\ - \Pr\left[Q\text{-}\mathbf{KEYDIST}^{\mathcal{A}}_{1,\mathsf{CMT}}(\lambda) \Rightarrow 1\right]|, \end{aligned}$$

  where games $\mathbf{KEYDIST}_0, \mathbf{KEYDIST}_1$ *are defined in Figure 3.*

- ***Statistically Binding.*** *There exists some (unbounded) algorithm* Ext*, such that for every (unbounded) algorithm* $\mathcal{A}$ *the following probability is at most* ε_{b}*:*

$$\Pr\left[\begin{array}{c|l} \mathsf{Com}(\mathsf{ck}, R'; \varphi') = \mathsf{com} & \mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda), \\ \wedge\ \ R \neq R' & \mathsf{ck} \leftarrow \mathsf{BGen}(\mathsf{par}),\ (\mathsf{com}, St) \leftarrow \mathcal{A}(\mathsf{ck}), \\ & R \leftarrow \mathsf{Ext}(\mathsf{ck}, \mathsf{com}),\ (R', \varphi') \leftarrow \mathcal{A}(St) \end{array}\right].$$

3.2 Our Construction with Key Aggregation

In this section, we construct a two-round multi-signature scheme with key aggregation. Although the scheme will not be tight, the security proof will not use rewinding, leading to an acceptable security loss. For our scheme, we need a

lossiness admitting linear function family $\mathsf{LF} = (\mathsf{LF.Gen}, \mathsf{F})$. It should also satisfy aggregation lossy soundness. Further, let $\mathsf{CMT} = (\mathsf{BGen}, \mathsf{TGen}, \mathsf{Com}, \mathsf{TCom}, \mathsf{TCol})$ be an $(\varepsilon_{\mathtt{b}}, \varepsilon_{\mathtt{g}}, \varepsilon_{\mathtt{t}})$-special commitment scheme for LF with key space $\mathcal{K}$ randomness space $\mathcal{G}$ and commitment space $\mathcal{H}$. We make use of random oracles $\mathsf{H}\colon \{0,1\}^* \to \mathcal{K}$, $\mathsf{H}_a\colon \{0,1\}^* \to \mathcal{S}$, and $\mathsf{H}_c\colon \{0,1\}^* \to \mathcal{S}$. We give a verbal description of our scheme $\mathsf{MS}_a[\mathsf{LF}, \mathsf{CMT}]$. Formally, the scheme is presented in **??**.

Setup and Key Generation. The public parameters of the scheme are $\mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda)$ defining the linear function $\mathsf{F} = \mathsf{F}(\mathsf{par}, \cdot)$. To generate a key (algorithm Gen), a user samples $\mathsf{sk} := x \xleftarrow{\$} \mathcal{D}$. The public key is $\mathsf{pk} := X := \mathsf{F}(x)$.

Key Aggregation. For N users with public keys $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$, the aggregated public key $\tilde{\mathsf{pk}}$ is computed (by algorithm Agg) as

$$\tilde{\mathsf{pk}} := \tilde{X} := \sum_{i=1}^{N} a_i \cdot X_i,$$

where $\mathsf{pk}_i = X_i$ and $a_i := \mathsf{H}_a(\langle \mathcal{P} \rangle, \mathsf{pk}_i)$ for each $i \in [N]$.

Signing Protocol. Suppose N users with public keys $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$ want to sign a message $\mathsf{m} \in \{0,1\}^*$. We describe the signing protocol (algorithms Sig_0, $\mathsf{Sig}_1, \mathsf{Sig}_2$) from the perspective of the first user, which holds a secret key $\mathsf{sk}_1 = x_1$ for public key $\mathsf{pk}_1 = X_1$.

1. *Commitment Phase.* The user derives the aggregated public key $\tilde{\mathsf{pk}}$ as described above. Then, it derives a commitment key $\mathsf{ck} := \mathsf{H}(\tilde{\mathsf{pk}}, \mathsf{m})$ depending on the message. The user samples an element $r_1 \xleftarrow{\$} \mathcal{D}$ and sets $R_1 := \mathsf{F}(r_1)$. Next, it commits to R_1 by sampling $\varphi_1 \xleftarrow{\$} \mathcal{G}$ and setting $\mathsf{com}_1 := \mathsf{Com}(\mathsf{ck}, R_1; \varphi_1)$. Finally, it sends $\mathsf{pm}_{1,1} := \mathsf{com}_1$ to all users.
2. *Response Phase.* Let $\mathcal{M}_1 = (\mathsf{pm}_{1,1}, \ldots, \mathsf{pm}_{1,N})$ be the list of messages output in the commitment phase. Here, message $\mathsf{pm}_{1,i}$ is sent by user i and has the form $\mathsf{pm}_{1,i} = \mathsf{com}_i$. With this notation, the user aggregates the commitments via $\mathsf{com} := \bigotimes_{i \in [N]} \mathsf{com}_i$. It computes the challenge c and coefficient a_1 via $c := \mathsf{H}_c(\tilde{\mathsf{pk}}, \mathsf{com}, \mathsf{m})$ and $a_1 := \mathsf{H}_a(\langle \mathcal{P} \rangle, \mathsf{pk}_1)$. Then, it computes the response s_1 as $s_1 := c \cdot a_1 \cdot x_1 + r_1$.
   Finally, the user sends $\mathsf{pm}_{2,1} := (s_1, \varphi_1)$ to all users.
3. *Aggregation Phase.* Let $\mathcal{M}_2 = (\mathsf{pm}_{2,1}, \ldots, \mathsf{pm}_{2,N})$ be the list of messages output in the response phase. Here, message $\mathsf{pm}_{2,i}$ is sent by user i and has the form $\mathsf{pm}_{2,i} = (s_i, \varphi_i)$. To compute the final signature, users aggregate the responses and commitment randomness as follows:

$$s := \sum_{i \in [N]} s_i, \;\; \varphi := \bigoplus_{i \in [N]} \varphi_i.$$

   They output the final signature $\sigma := (\mathsf{com}, s, \varphi)$.

Verification. For verification (algorithm Ver), let $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$ be a multi-set of public keys, $\mathsf{m} \in \{0,1\}^*$ be a message, and $\sigma = (\mathsf{com}, s, \varphi)$ be a signature. To verify σ, we determine the aggregated public key $\tilde{\mathsf{pk}} = \tilde{X}$ as above. We reconstruct the commitment key $\mathsf{ck} := \mathsf{H}(\tilde{\mathsf{pk}}, \mathsf{m})$, and the challenge $c := \mathsf{H}_c(\tilde{\mathsf{pk}}, \mathsf{com}, \mathsf{m})$. Then, we output 1 if and only if the following equation holds:

$$\mathsf{com} = \mathsf{Com}\left(\mathsf{ck}, \mathsf{F}(s) - c \cdot \tilde{X}; \varphi\right).$$

Completeness easily follows from the homomorphic properties of CMT and F. For a similar calculation, we refer to the proof of Lemma 2.

Lemma 1. *Let* LF *be a linear function family. Let* CMT *be a* $(\varepsilon_{\mathsf{b}}, \varepsilon_{\mathsf{g}}, \varepsilon_{\mathsf{t}})$*-special commitment scheme for* LF*. Then* $\mathsf{MS}_a[\mathsf{LF}, \mathsf{CMT}]$ *is complete.*

Theorem 1. *Let* LF *be a* ε_{l}*-lossiness admitting linear function family with* $\varepsilon_{\mathsf{al}}$*-aggregation lossy soundness. Let* CMT *be a* $(\varepsilon_{\mathsf{b}}, \varepsilon_{\mathsf{g}}, \varepsilon_{\mathsf{t}})$*-special commitment scheme for* LF*. Further, let* $\mathsf{H}\colon \{0,1\}^* \to \mathcal{K}, \mathsf{H}_a\colon \{0,1\}^* \to \mathcal{S}$*, and* $\mathsf{H}_c\colon \{0,1\}^* \to \mathcal{S}$ *be random oracles. Then* $\mathsf{MS}_a[\mathsf{LF}, \mathsf{CMT}]$ *is* $\mathsf{MS\text{-}EUF\text{-}CMA}$ *secure.*

Concretely, for any PPT algorithm $\mathcal{A}$ *that makes at most* $Q_{\mathsf{H}}, Q_{\mathsf{H}_a}, Q_{\mathsf{H}_c}, Q_S$ *queries to oracles* $\mathsf{H}, \mathsf{H}_a, \mathsf{H}_c, \textsc{Sig}_0$*, respectively, there are PPT algorithms* $\mathcal{B}, \mathcal{B}'$ *with* $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{A}), \mathbf{T}(\mathcal{B}') \approx \mathbf{T}(\mathcal{A})$ *and*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{MS\text{-}EUF\text{-}CMA}}_{\mathcal{A},\mathsf{MS}_a[\mathsf{LF},\mathsf{CMT}]}(\lambda) \leq\ & \varepsilon_{\mathsf{g}} + 4Q_S^2\varepsilon_{\mathsf{t}} + 4Q_S\varepsilon_{\mathsf{g}} + 4Q_S Q_{\mathsf{H}} Q_{\mathsf{H}_c}\varepsilon_{\mathsf{b}} \\ & + \frac{4Q_S}{|\mathcal{R}|} + \frac{4Q_S Q_{\mathsf{H}_a} Q_{\mathsf{H}_c}}{|\mathcal{S}|} + 4Q_S Q_{\mathsf{H}_a} Q_{\mathsf{H}_c}\varepsilon_{\mathsf{al}} \\ & + 4Q_S\left(\mathsf{Adv}^{Q_{\mathsf{H}}\text{-}\mathsf{keydist}}_{\mathcal{B},\mathsf{CMT}}(\lambda) + \mathsf{Adv}^{\mathsf{keydist}}_{\mathcal{B}',\mathsf{LF}}(\lambda)\right).\end{aligned}$$

We postpone the proof to the full version [34].

3.3 Our Tight Construction

In this section, we present a tightly secure two-round multi-signature scheme $\mathsf{MS}_t[\mathsf{LF}, \mathsf{CMT}] = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$. Let us first describe the building blocks that we need. We make use of a lossiness admitting linear function family $\mathsf{LF} = (\mathsf{LF.Gen}, \mathsf{F})$. Also, let $\mathsf{CMT} = (\mathsf{BGen}, \mathsf{TGen}, \mathsf{Com}, \mathsf{TCom}, \mathsf{TCol})$ be an $(\varepsilon_{\mathsf{b}}, \varepsilon_{\mathsf{g}}, \varepsilon_{\mathsf{t}})$-special commitment scheme for LF with key space $\mathcal{K}$ randomness space $\mathcal{G}$ and commitment space $\mathcal{H}$. We make use of random oracles $\mathsf{H}\colon \{0,1\}^* \to \mathcal{K}, \mathsf{H}_b\colon \{0,1\}^* \to \{0,1\}$, and $\mathsf{H}_c\colon \{0,1\}^* \to \mathcal{S}$. We give a verbal description of the scheme. Formally, the scheme is presented in **??**.

Setup and Key Generation. The public parameters of the scheme are $\mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda)$. They define the linear function $\mathsf{F} = \mathsf{F}(\mathsf{par}, \cdot)$. To generate a key (algorithm Gen), a user samples $x_0, x_1 \xleftarrow{\$} \mathcal{D}$ and a seed $\mathsf{seed} \xleftarrow{\$} \{0,1\}^\lambda$. Then, it sets

$$\mathsf{sk} := (x_0, x_1, \mathsf{seed}), \quad \mathsf{pk} := (X_0, X_1) := (\mathsf{F}(x_0), \mathsf{F}(x_1)).$$

Signing Protocol. Suppose N users with public keys $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$ want to sign a message $\mathsf{m} \in \{0,1\}^*$. We describe the signing protocol (algorithms $\mathsf{Sig}_0, \mathsf{Sig}_1, \mathsf{Sig}_2$) from the perspective of the first user, which holds a secret key $\mathsf{sk}_1 = (x_{1,0}, x_{1,1}, \mathsf{seed}_1)$ for public key $\mathsf{pk}_1 = (X_{1,0}, X_{1,1})$.

1. *Commitment Phase.* The user derives commitment keys $\mathsf{ck}_0 := \mathsf{H}(0, \langle\mathcal{P}\rangle, \mathsf{m})$, $\mathsf{ck}_1 := \mathsf{H}(1, \langle\mathcal{P}\rangle, \mathsf{m})$ depending on the message. Then, the user computes a bit $b_1 := \mathsf{H}_b(\mathsf{seed}_1, \langle\mathcal{P}\rangle, \mathsf{m})$. It samples two elements $r_{1,0}, r_{1,1} \xleftarrow{\$} \mathcal{D}$ and sets

$$R_{1,0} := \mathsf{F}(r_{1,0}), \quad R_{1,1} := \mathsf{F}(r_{1,1}).$$

   Next, it commits to the resulting elements by sampling $\varphi_{1,0}, \varphi_{1,1} \xleftarrow{\$} \mathcal{G}$ and setting

$$\mathsf{com}_{1,0} := \mathsf{Com}(\mathsf{ck}_0, R_{1,0}; \varphi_{1,0}), \quad \mathsf{com}_{1,1} := \mathsf{Com}(\mathsf{ck}_1, R_{1,1}; \varphi_{1,1}).$$

   Finally, it sends $\mathsf{pm}_{1,1} := (b_1, \mathsf{com}_{1,0}, \mathsf{com}_{1,1})$ to all users.
2. *Response Phase.* Let $\mathcal{M}_1 = (\mathsf{pm}_{1,1}, \ldots, \mathsf{pm}_{1,N})$ be the list of messages output in the commitment phase. Here, message $\mathsf{pm}_{1,i}$ is sent by user i and has the form $\mathsf{pm}_{1,i} = (b_i, \mathsf{com}_{i,0}, \mathsf{com}_{i,1})$. With this notation, the user sets $B := b_1 \ldots b_N \in \{0,1\}^N$. Then, it aggregates the commitments via

$$\mathsf{com}_0 := \bigotimes_{i \in [N]} \mathsf{com}_{i,0}, \quad \mathsf{com}_1 := \bigotimes_{i \in [N]} \mathsf{com}_{i,1}.$$

   It computes user specific challenges via

$$c_{1,0} := \mathsf{H}_c(\mathsf{pk}_1, \mathsf{com}_0, \mathsf{m}, \langle\mathcal{P}\rangle, B, 0), \quad c_{1,1} := \mathsf{H}_c(\mathsf{pk}_1, \mathsf{com}_1, \mathsf{m}, \langle\mathcal{P}\rangle, B, 1),$$

   and the responses as

$$s_{1,0} := c_{1,0} \cdot x_{1,b_1} + r_{1,0}, \quad s_{1,1} := c_{1,1} \cdot x_{1,1-b_1} + r_{1,1}.$$

   Observe that the bit b_1 determines the link between the responses, challenges, and public keys. Finally, the user sends $\mathsf{pm}_{2,1} := (s_{1,0}, s_{1,1}, \varphi_{1,0}, \varphi_{1,1})$ to all users.
3. *Aggregation Phase.* Let $\mathcal{M}_2 = (\mathsf{pm}_{2,1}, \ldots, \mathsf{pm}_{2,N})$ be the list of messages output in the response phase. Here, message $\mathsf{pm}_{2,i}$ is sent by user i and has the form $\mathsf{pm}_{2,i} = (s_{i,0}, s_{i,1}, \varphi_{i,0}, \varphi_{i,1})$. To compute the final signature, users aggregate the responses and commitment randomness as follows:

$$s_0 := \sum_{i \in [N]} s_{i,0}, \; s_1 := \sum_{i \in [N]} s_{i,1}, \; \varphi_0 := \bigoplus_{i \in [N]} \varphi_{i,0}, \; \varphi_1 := \bigoplus_{i \in [N]} \varphi_{i,1}.$$

   They define $\sigma_0 := (\mathsf{com}_0, \varphi_0, s_0), \sigma_1 := (\mathsf{com}_1, \varphi_1, s_1)$ and output the final signature $\sigma := (\sigma_0, \sigma_1, B)$.

Verification. For verification (algorithm Ver), let $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$ be a multi-set of public keys, $\mathsf{m} \in \{0,1\}^*$ be a message, and $\sigma = (\sigma_0, \sigma_1, B)$ be a signature. To verify σ, we write $B = b_1 \ldots b_N, \sigma_0 = (\mathsf{com}_0, \varphi_0, s_0)$ and $\sigma_1 = (\mathsf{com}_1, \varphi_1, s_1)$. Further, we write the public keys pk_i as $\mathsf{pk}_i = (X_{i,0}, X_{i,1})$. We reconstruct the commitment keys $\mathsf{ck}_0 := \mathsf{H}(0, \langle\mathcal{P}\rangle, \mathsf{m}), \mathsf{ck}_1 := \mathsf{H}(1, \langle\mathcal{P}\rangle, \mathsf{m})$, and the user specific challenges

$$c_{i,0} := \mathsf{H}_c(\mathsf{pk}_i, \mathsf{com}_0, \mathsf{m}, \langle\mathcal{P}\rangle, B, 0), \quad c_{i,1} := \mathsf{H}_c(\mathsf{pk}_i, \mathsf{com}_1, \mathsf{m}, \langle\mathcal{P}\rangle, B, 1).$$

Then, we output 1 if and only if the following two equations hold:

$$\mathsf{com}_0 = \mathsf{Com}\left(\mathsf{ck}_0, \mathsf{F}(s_0) - \sum_{i=1}^{N} c_{i,0} \cdot X_{i,b_i}; \varphi_0\right)$$
$$\mathsf{com}_1 = \mathsf{Com}\left(\mathsf{ck}_1, \mathsf{F}(s_1) - \sum_{i=1}^{N} c_{i,1} \cdot X_{i,1-b_i}; \varphi_1\right).$$

Lemma 2. *Let* LF *be a linear function family. Let* CMT *be a* $(\varepsilon_\mathsf{b}, \varepsilon_\mathsf{g}, \varepsilon_\mathsf{t})$*-special commitment scheme for* LF*. Then* $\mathsf{MS}_t[\mathsf{LF}, \mathsf{CMT}]$ *is complete.*

The proof is an easy calculation and is given in the full version [34].

Theorem 2. *Let* LF *be a* ε_l*-lossiness admitting linear function family. Let* CMT *be a* $(\varepsilon_\mathsf{b}, \varepsilon_\mathsf{g}, \varepsilon_\mathsf{t})$*-special commitment scheme for* LF*. Further, let* $\mathsf{H}\colon \{0,1\}^* \to \mathcal{K}$, $\mathsf{H}_b\colon \{0,1\}^* \to \{0,1\}, \mathsf{H}_c\colon \{0,1\}^* \to \mathcal{S}$ *be random oracles. Then* $\mathsf{MS}_t[\mathsf{LF}, \mathsf{CMT}]$ *is* MS-EUF-CMA *secure.*

Concretely, for any PPT algorithm $\mathcal{A}$ *that makes at most* $Q_\mathsf{H}, Q_{\mathsf{H}_b}, Q_{\mathsf{H}_c}, Q_S$ *queries to oracles* $\mathsf{H}, \mathsf{H}_b, \mathsf{H}_c, \textsc{Sig}_0$*, respectively, there are PPT algorithms* $\mathcal{B}, \mathcal{B}'$ *with* $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{A}), \mathbf{T}(\mathcal{B}') \approx \mathbf{T}(\mathcal{A})$ *and*

$$\mathsf{Adv}^{\mathsf{MS\text{-}EUF\text{-}CMA}}_{\mathcal{A},\mathsf{MS}_t[\mathsf{LF},\mathsf{CMT}]}(\lambda) \le \frac{Q_{\mathsf{H}_b}}{2^\lambda} + 4\varepsilon_\mathsf{g} + 2Q_S\varepsilon_\mathsf{t} + 2Q_\mathsf{H}Q_{\mathsf{H}_c}\varepsilon_\mathsf{b} + 2Q_{\mathsf{H}_c}\varepsilon_\mathsf{l} + 2 \cdot \mathsf{Adv}^{Q_\mathsf{H}\text{-}\mathsf{keydist}}_{\mathcal{B},\mathsf{CMT}}(\lambda) + 2 \cdot \mathsf{Adv}^{\mathsf{keydist}}_{\mathcal{B}',\mathsf{LF}}(\lambda).$$

Proof. Set $\mathsf{MS} := \mathsf{MS}_t[\mathsf{LF}, \mathsf{CMT}]$. Let $\mathcal{A}$ be a PPT algorithm as in the statement. We prove the claim via a sequence of games $\mathbf{G}_0$-$\mathbf{G}_8$. The games are formally presented in **??????**, and we describe and analyze them verbally. For each game $\mathbf{G}_i, i \in [8]$, we define

$$\mathsf{Adv}_i := \Pr[\mathbf{G}_i \Rightarrow 1].$$

Game $\mathbf{G}_0$: We define $\mathbf{G}_0$ to be exactly as $\mathbf{MS\text{-}EUF\text{-}CMA}^{\mathcal{A}}_{\mathsf{MS}}$, with the following modification: The adversary $\mathcal{A}$ does not get access to oracle $\textsc{Sig}_2$. Note that in MS, algorithm Sig_2 does not make any use of the secret key or a secret state and can be publicly run using the messages output in Sig_0 and Sig_1. Therefore, for any adversary in the original game, there is an adversary in game $\mathbf{G}_0$ that simulates oracle $\textsc{Sig}_2$ and has the same advantage.

Before we proceed, let us describe game $\mathbf{G}_0$ in more detail to fix some notation. In the beginning, the game samples parameters $\mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda)$. It also samples a public key $\mathsf{pk}^* = (X_{1,0}, X_{1,1}) = (\mathsf{F}(x_{1,0}), \mathsf{F}(x_{1,1}))$ for a secret key $\mathsf{sk}^* = (x_{1,0}, x_{1,1}, \mathsf{seed}_1)$ with $x_{1,0}, x_{1,1} \xleftarrow{\$} \mathcal{D}, \mathsf{seed}_1 \xleftarrow{\$} \{0,1\}^\lambda$. Then, it runs $\mathcal{A}$ on input $\mathsf{par}, \mathsf{pk}^*$ with access to the following oracles:

- Signing oracles $\textsc{Sig}_0, \textsc{Sig}_1$: The oracles simulate algorithms Sig_0 and Sig_1 on secret key sk^*, respectively. Here, $\mathcal{A}$ can submit a query $\textsc{Sig}_0(\mathcal{P}, \mathsf{m})$ to start a new interaction in which message m is signed for public keys $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$. We assume that $\mathsf{pk}^* = \mathsf{pk}_1$, and the oracle adds $(\mathcal{P}, \mathsf{m})$ to a list $\mathcal{L}$.
- Random oracles $\mathsf{H}, \mathsf{H}_b, \mathsf{H}_c$: The random oracles H, H_c are simulated honestly via lazy sampling. To this end, the game holds maps h, h_c that map the inputs of the respective random oracles to their outputs. Random oracle H_b, however, is simulated by forwarding the query to an internal oracle $\bar{\mathsf{H}}_b$ with the same interface. This oracle holds a similar map $\hat{h}_b$, is kept internally by the game, and is not provided to the adversary. Looking ahead, this indirection allows us to distinguish queries to H_b that some of the following games issue from the queries that the adversary issues.

In the end, $\mathcal{A}$ outputs a forgery $(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$. The game outputs 1 if and only if $\mathsf{pk}^* \in \mathcal{P}^*$,$(\mathcal{P}^*, \mathsf{m}^*) \notin \mathcal{L}$, and $\mathsf{Ver}(\mathcal{P}^*, \mathsf{m}^*, \sigma^*) = 1$. Without loss of generality, we assume that the public key pk^* is equal to pk_1 for $\mathcal{P}^* = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$. To fix notation, write $\sigma^* = (\sigma_0^*, \sigma_1^*, B^*)$, $B^* = b_1^* \ldots b_N^*$ and $\sigma_0^* = (\mathsf{com}_0^*, \varphi_0^*, s_0^*), \sigma_1^* = (\mathsf{com}_1^*, \varphi_1^*, s_1^*)$. Clearly, we have

$$\mathsf{Adv}_0 = \mathsf{Adv}^{\mathsf{MS\text{-}EUF\text{-}CMA}}_{\mathcal{A},\mathsf{MS}_t[\mathsf{LF},\mathsf{CMT}]}(\lambda).$$

Game $\mathbf{G}_1$: In game $\mathbf{G}_1$, we add an abort. Namely, the game sets $\mathsf{bad} := 1$, and aborts, if the adversary makes a random oracle query $\mathsf{H}_b(\mathsf{seed}_1, \cdot)$. Note that this does not include the queries that are made by the game itself, as these are done using oracle $\bar{\mathsf{H}}_b$ directly. As the only information about seed_1 that $\mathcal{A}$ gets are the values of $\mathsf{H}_b(\mathsf{seed}_1, \cdot)$, and seed_1 is sampled uniformly at random from $\{0,1\}^\lambda$, we can upper bound the probability of bad by $Q_{\mathsf{H}_b}/2^\lambda$. Therefore, we have

$$|\mathsf{Adv}_0 - \mathsf{Adv}_1| \leq \Pr[\mathsf{bad}] \leq \frac{Q_{\mathsf{H}_b}}{2^\lambda}.$$

Game $\mathbf{G}_2$: In game $\mathbf{G}_2$, we restrict the winning condition. Namely, the game outputs 0, if the forgery $(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$ output by $\mathcal{A}$ satisfies $b_1^* \neq 1 - \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P}^* \rangle, \mathsf{m}^*)$. Recall that b_1^* is the bit related to $\mathsf{pk}_1 = \mathsf{pk}^*$ that is included in the signature σ^*. Assuming $\mathbf{G}_1$ outputs 1, we know that $(\mathcal{P}^*, \mathsf{m}^*) \notin \mathcal{L}$. Therefore, $\mathcal{A}$ can only get information about the bit $\bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P}^* \rangle, \mathsf{m}^*)$, if it queries the wrapper random oracle H_b at this position. However, in this case $\mathbf{G}_1$ would set $\mathsf{bad} := 1$ and abort. Thus, the view of $\mathcal{A}$ is independent of bit $\bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P}^* \rangle, \mathsf{m}^*)$. We obtain

$$\mathsf{Adv}_2 = \Pr[\mathbf{G}_2 \Rightarrow 1] = \Pr\left[\mathbf{G}_1 \Rightarrow 1 \wedge b_1^* = 1 - \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P}^* \rangle, \mathsf{m}^*)\right] = \frac{1}{2}\mathsf{Adv}_1.$$

Game $\mathbf{G}_3$: In game $\mathbf{G}_3$, the game aborts if $(\mathsf{par}, x_{1,1}) \notin \mathsf{Good}$, where Good is as in the definition of a special commitment scheme. It is clear that

$$|\mathsf{Adv}_2 - \mathsf{Adv}_3| \leq \Pr\left[(\mathsf{par}, x_{1,1}) \notin \mathsf{Good}\right] \leq \varepsilon_{\mathsf{g}}.$$

Game $\mathbf{G}_4$: In game $\mathbf{G}_4$, we change the behavior of random oracle H. Recall that before, to answer a query $\mathsf{H}(b, \langle \mathcal{P} \rangle, \mathsf{m})$ for which the hash value has not been defined, a key $\mathsf{ck} \xleftarrow{\$} \mathcal{K}$ was sampled and returned. In this game, the oracle instead distinguishes two cases. In the first case, if $b = 1 - \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P} \rangle, \mathsf{m})$, the game samples $(\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{TGen}(\mathsf{par}, X_{1,1})$. It also stores $tr[\langle \mathcal{P} \rangle, \mathsf{m}] := \mathsf{td}$, where tr is a map. In the second case, if $b = \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P} \rangle, \mathsf{m})$, it samples $\mathsf{ck} \leftarrow \mathsf{BGen}(\mathsf{par})$. In both cases, ck is returned as before. To see that $\mathbf{G}_3$ and $\mathbf{G}_4$ are indistinguishable, we first note that for the first case, the distribution of ck stays the same. This is because we can assume $(\mathsf{par}, x_{1,1}) \in \mathsf{Good}$ due to the previous change. The keys returned in the second case are indistinguishable by the multi-key indistinguishability of CMT. More precisely, we give a reduction $\mathcal{B}$ against the multi-key indistinguishability of CMT that interpolates between $\mathbf{G}_3$ and $\mathbf{G}_4$. The reduction gets as input $\mathsf{par}, x_{1,1}$ and Q_{H} commitment keys $\mathsf{ck}_1, \ldots, \mathsf{ck}_{Q_{\mathsf{H}}}$. It simulates $\mathbf{G}_3$ for $\mathcal{A}$ with par while embedding the commitment keys in random oracle responses for queries $\mathsf{H}(b, \langle \mathcal{P} \rangle, \mathsf{m})$ with $b = 1 - \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P} \rangle, \mathsf{m})$. In the end, it outputs whatever the game outputs[7] . We have

$$|\mathsf{Adv}_3 - \mathsf{Adv}_4| \leq \mathsf{Adv}^{Q_{\mathsf{H}}\text{-}\mathsf{keydist}}_{\mathcal{B},\mathsf{CMT}}(\lambda).$$

Game $\mathbf{G}_5$: In game $\mathbf{G}_5$, we change the signing oracles $\textsc{Sig}_0, \textsc{Sig}_1$. Our goal is to eliminate the use of the secret key component $x_{1,1}$. Recall that in previous games, oracle $\textsc{Sig}_0$ derived a bit $b_1 := \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P} \rangle, \mathsf{m})$ and sampled random $r_{1,0}, r_{1,1}$ and $\varphi_{1,0}, \varphi_{1,1}$. Then, these were used with to compute commitments $\mathsf{com}_{1,0}, \mathsf{com}_{1,1}$, which where then output together with b_1. Then, in oracle $\textsc{Sig}_1$ the values $s_{1,0}, s_{1,1}$ were computed using the secret keys $x_{1,b_1}, x_{1,1-b_1}$, respectively.

In this game, we change how the commitment $\varphi_{1,1-b_1}$ and the value $s_{1,1-b_1}$ is computed to eliminate the dependency on $x_{1,1}$. Namely, in oracle $\textsc{Sig}_0$, we do not compute $r_{1,1-b_1}, \varphi_{1,1-b_1}$ and $R_{1,1-b_1}$ anymore. Instead, we compute the commitment $\mathsf{com}_{1,1-b_1}$ via

$$\mathsf{td} := tr[\langle \mathcal{P} \rangle, , \mathsf{m}], \quad (\mathsf{com}_{1,1-b_1}, St) \leftarrow \mathsf{TCom}(\mathsf{ck}_{1-b_1}, \mathsf{td}).$$

Note that $\mathsf{ck}_{1-b_1} = \mathsf{H}(1 - b_1, \langle \mathcal{P} \rangle, \mathsf{m})$, and therefore ck_{1-b_1} and td were generated using $\mathsf{TGen}(\mathsf{par}, X_{1,1})$ due to the change in $\mathbf{G}_4$. Later, in oracle $\textsc{Sig}_1$, we derive

$$(\varphi_{1,1-b_1}, R_{1-b_1}, s_{1,1-b_1}) \leftarrow \mathsf{TCol}(St, c_{1,1-b_1}).$$

Then, message $\mathsf{pm}_{2,1} := (s_{1,0}, s_{1,1}, \varphi_{1,0}, \varphi_{1,1})$ is output as before.

[7] Note that at this point, it was important that we introduced the oracle $\bar{\mathsf{H}}_b$. This is because otherwise, if we queried $\mathsf{H}_b(\mathsf{seed}_1, \cdot)$ in oracle H, game $\mathbf{G}_3$ would always output 0 and the games would not be indistinguishable.

We can easily argue indistinguishability by using the special trapdoor property of CMT Q_{S_0} many times and get

$$|\mathsf{Adv}_4 - \mathsf{Adv}_5| \leq Q_S \varepsilon_{\mathtt{t}}.$$

Game $\mathbf{G}_6$: Here we do not abort if $(\mathsf{par}, x_{1,1}) \notin \mathsf{Good}$ anymore. That is, we revert the change introduced in $\mathbf{G}_3$. It is clear that

$$|\mathsf{Adv}_5 - \mathsf{Adv}_6| \leq \Pr\left[(\mathsf{par}, x_{1,1}) \notin \mathsf{Good}\right] \leq \varepsilon_{\mathtt{g}}.$$

Game $\mathbf{G}_7$: In game $\mathbf{G}_7$, we change how the public key component $X_{1,1}$ is computed. Recall that before, $X_{1,1}$ is computed as $X_{1,1} := \mathsf{F}(x_{1,1})$ for $x_{1,1} \xleftarrow{\$} x_{1,1} \xleftarrow{\$} \mathcal{D}$. Also, note that due to the previous changes, the value $x_{1,1}$ is not used anymore. In $\mathbf{G}_7$, we sample $X_{1,1} \xleftarrow{\$} \mathcal{R}$. A direct reduction $\mathcal{B}'$ against the key indistinguishability of the lossiness admitting linear function family LF shows indistinguishability of $\mathbf{G}_6$ and $\mathbf{G}_7$. Concretely, $\mathcal{B}'$ gets par and $X_{1,1}$ as input, and simulates $\mathbf{G}_6$ for $\mathcal{A}$. In the end, it outputs whatever the game outputs. We have

$$|\mathsf{Adv}_6 - \mathsf{Adv}_7| \leq \mathsf{Adv}^{\mathsf{keydist}}_{\mathcal{B}',\mathsf{LF}}(\lambda).$$

Game $\mathbf{G}_8$: In game $\mathbf{G}_8$, we change how H_c is executed. Concretely, consider a query $\mathsf{H}_c(\mathsf{pk}, \mathsf{com}, \mathsf{m}, \langle \mathcal{P} \rangle, B, b)$ with $\mathsf{pk} = \mathsf{pk}^*$ and $b = \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P} \rangle, \mathsf{m})$. For these queries, the game now runs $R \leftarrow \mathsf{Ext}(\mathsf{H}(b, \langle \mathcal{P} \rangle, \mathsf{m}), \mathsf{com})$ and stores $r[\mathsf{com}, \mathsf{m}, \langle \mathcal{P} \rangle, B] := R$, where r is another map. Here, Ext is the (unbounded) extractor for the statistical binding property of CMT. The rest of the oracle does not change. Note that for b of this form, the value $\mathsf{ck} = \mathsf{H}(b, \langle \mathcal{P} \rangle, \mathsf{m})$ is sampled as $\mathsf{ck} \leftarrow \mathsf{BGen}(\mathsf{par})$ (cf. $\mathbf{G}_4$). We also slightly change the winning condition of the game. Namely, in $\mathbf{G}_8$, consider the forgery $(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$ with $\sigma^* = (\sigma_0^*, \sigma_1^*, B^*)$, $B^* = b_1^* \dots b_N^*$, and let $R_0^*, R_1^* \in \mathcal{R}$ be the values that are computed during the execution of $\mathsf{Ver}(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$. The game returns 0 if $R^*_{1-b_1^*} \neq r[\mathsf{com}^*_{1-b_1^*}, \mathsf{m}^*, \langle \mathcal{P}^* \rangle, B^*]$.

We claim that indistinguishability of $\mathbf{G}_7$ and $\mathbf{G}_8$ can be argued using the statistical binding property of CMT. To see this, assume that $\mathbf{G}_7$ outputs 1. Then, due to the change in $\mathbf{G}_2$, we know that $1 - b_1^* = \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P}^* \rangle, \mathsf{m}^*)$. Therefore, in the corresponding query $\mathsf{H}_c(\mathsf{pk}_1, \mathsf{com}^*_{1-b_1^*}, \mathsf{m}^*, \langle \mathcal{P}^* \rangle, B^*, 1 - b_1^*)$ algorithm Ext was run and the value $r[\mathsf{com}^*_{1-b_1^*}, \mathsf{m}^*, \langle \mathcal{P}^* \rangle, B^*]$ is defined. Next, by definition of Ver, we have $\mathsf{Com}(\mathsf{ck}_{1-b_1^*}, R^*_{1-b_1^*}; \varphi^*_{1-b_1^*}) = \mathsf{com}^*_{1-b_1^*}$. Therefore, if $R^*_{1-b_1^*} \neq r[\mathsf{com}^*_{1-b_1^*}, \mathsf{m}^*, \langle \mathcal{P}^* \rangle, B^*]$, we have a contradiction to the statistical binding property of CMT. More precisely, we sketch an (unbounded) reduction from the statistical binding property. Namely, this reduction gets as input par and a commitment key ck^*. Then, the reduction guesses $i_{\mathsf{H}} \xleftarrow{\$} [Q_{\mathsf{H}}]$ and $i_{\mathsf{H}_c} \xleftarrow{\$} [Q_{\mathsf{H}_c}]$. It simulates game $\mathbf{G}_8$ honestly, except for query i_{H} to random oracle H and query i_{H_c} to random oracle H_c. If it had to sample a $\mathsf{ck} \leftarrow \mathsf{BGen}(\mathsf{par})$ in the former query, it instead responds with ck^*. Similarly, if it had to run Ext in the latter query, it outputs com to the binding experiment. If these queries are the queries of interest (i.e. query i_{H} was used to derive $\mathsf{ck}_{1-b_1^*}$ and query i_{H_c} was used to

derive $c^*_{1,1-b^*_1}$) for the forgery, and $R^*_{1-b^*_1} \neq r[\mathsf{com}^*_{1-b^*_1}, \mathsf{m}^*, \langle \mathcal{P}^* \rangle, B^*]$, then the reduction outputs $R^*_{1-b^*_1}; \varphi^*_{1-b^*_1}$. Otherwise, it outputs $\perp$. It is easy to see that if the reduction guesses the correct queries and the bad event separating $\mathbf{G}_7$ and $\mathbf{G}_8$ occurs, then it breaks the statistical binding property. As the view of $\mathcal{A}$ is as in $\mathbf{G}_8$, and independent of $(i_{\mathsf{H}}, i_{\mathsf{H}_c})$, we obtain

$$|\mathsf{Adv}_7 - \mathsf{Adv}_8| \leq Q_{\mathsf{H}} Q_{\mathsf{H}_c} \varepsilon_{\mathsf{b}}.$$

Finally, we use lossy soundness of LF to bound the probability that $\mathbf{G}_8$ outputs 1. To do that, we give an unbounded reduction from the lossy soundness experiment, which is as follows.

- The reduction gets $\mathsf{par}, X_{1,1}$ as input. It samples $\hat{i} \xleftarrow{\$} [Q_{\mathsf{H}_c}]$. Then, it simulates $\mathbf{G}_8$ honestly until $\mathcal{A}$ outputs a forgery, except for query $\hat{i}$ to oracle H_c.
- Consider this query $\mathsf{H}_c(\mathsf{pk}, \mathsf{com}, \mathsf{m}, \langle \mathcal{P} \rangle, B, b)$. The reduction aborts its execution, if the hash value for this query is already defined, or if $\mathsf{pk} \neq \mathsf{pk}^* \vee b \neq \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P} \rangle, \mathsf{m})$. Otherwise, it runs $\hat{R} \leftarrow \mathsf{Ext}(\mathsf{H}(b, \langle \mathcal{P} \rangle, \mathsf{m}), \mathsf{com})$ as in $\mathbf{G}_8$. Then, it parses $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$ and $B = b_1 \ldots b_N$. It parses $\mathsf{pk}_i = (X_{i,0}, X_{i,1})$ for each $i \in [N]$, and it sets $c_{i,b} = \mathsf{H}_c(\mathsf{pk}_i, \mathsf{com}, \mathsf{m}, \langle \mathcal{P} \rangle, B, b)$ for each $i \in [N] \setminus \{1\}$. Next, it defines

$$R := \hat{R} + \sum_{i=2}^{N} c_{i,b} \cdot X_{i,\hat{b}_i},$$

  where $\hat{b}_i := (b + b_i) \mod 2$. It outputs R to the lossy soundness game and obtains a value c in return. Then, it sets $h_c[\mathsf{pk}, \mathsf{com}, \mathsf{m}, \langle \mathcal{P} \rangle, B, b] := c$ and continues the simulation as in $\mathbf{G}_8$.
- When the reduction gets the forgery $(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$ from $\mathcal{A}$, it runs all the verification steps in $\mathbf{G}_8$. Additionally, it checks if the value $\mathsf{H}_c(\mathsf{pk}_1, \mathsf{com}^*_{1-b^*_1}, \mathsf{m}^*, \langle \mathcal{P}^* \rangle, B^*, 1 - b^*_1)$ was defined during query $\hat{i}$ to H_c. If this is not the case, it aborts its execution. Otherwise, it returns $s := s^*_{1-b^*_1}$ to the lossy soundness game.

It is clear that the view of $\mathcal{A}$ is independent of the index $\hat{i}$ until a potential abort of the reduction. Also, if the reduction does not abort its execution, it perfectly simulates game $\mathbf{G}_8$ for $\mathcal{A}$. Thus, it remains to show that if $\mathbf{G}_8$ outputs 1, then the values output by the reduction satisfy $\mathsf{F}(s) - c \cdot X_{1,1} = R$. Once we have shown this, it follows that

$$\mathsf{Adv}_8 \leq Q_{\mathsf{H}_c} \varepsilon_{\mathsf{l}}.$$

To show the desired property, assume that the reduction does not abort and $\mathbf{G}_8$ outputs 1. Then, define $\hat{b}^*_i = (1 - b^*_1 + b_i) \mod 2$ for all $i \in [N]$. Note that $\hat{b}^*_i = 1$. Due to the change in $\mathbf{G}_2$, we have

$$b = 1 - b^*_1 = \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P}^* \rangle, \mathsf{m}^*).$$

As the reduction guessed the right query and does not abort, we have

$$c^*_{1,1-b^*_1} = \mathsf{H}_c(\mathsf{pk}_1, \mathsf{com}^*_{1-b^*_1}, \mathsf{m}^*, \langle \mathcal{P}^* \rangle, B^*, 1 - b^*_1) = c.$$

Due to the change in $\mathbf{G}_8$, we have

$$\mathsf{F}(s^*_{1-b^*_1}) - \sum_{i=1}^{N} c^*_{i,1-b^*_1} \cdot X_{i,\hat{b}^*_i} = R^*_{1-b^*_1} = \hat{R}.$$

Therefore, we have

$$\begin{aligned}\mathsf{F}(s) - c \cdot X_{1,1} &= \mathsf{F}(s^*_{1-b^*_1}) - c^*_{1,1-b^*_1} \cdot X_{1,1} \\ &= \mathsf{F}(s^*_{1-b^*_1}) - \sum_{i=1}^{N} c^*_{i,1-b^*_1} \cdot X_{i,\hat{b}^*_i} + \sum_{i=2}^{N} c^*_{i,1-b^*_1} \cdot X_{i,\hat{b}^*_i} \\ &= \hat{R} + \sum_{i=2}^{N} c^*_{i,1-b^*_1} \cdot X_{i,\hat{b}^*_i} = R.\end{aligned}$$

Concluded. □

4 Instantiation

In this section, we show how to instantiate the building blocks that are needed for our constructions in the previous section. Concretely, we give a linear function family and a commitment scheme based on the DDH assumption. Then, we also discuss the efficiency of the resulting multi-signature schemes.

4.1 Linear Function Family

We make use of the well-known [27] linear function family $\mathsf{LF}_{\mathsf{DDH}} = (\mathsf{Gen}, \mathsf{F})$ based on the DDH assumption. Precisely, let GGen be an algorithm that on input 1^λ outputs the description of a prime order group $\mathbb{G}$ of order p with generator g. Then, Gen runs GGen and outputs[8] $\mathsf{par} := (g, h) \in \mathbb{G}^2$ for $h \xleftarrow{\$} \mathbb{G}$. Then, the set of scalars, domain, range, and function $\mathsf{F}(\mathsf{par}, \cdot)$ are given as follows:

$$\mathcal{S} := \mathbb{Z}_p, \quad \mathcal{D} := \mathbb{Z}_p, \quad \mathcal{R} := \mathbb{G} \times \mathbb{G}, \quad \mathsf{F}(\mathsf{par}, x) := (g^x, h^x).$$

It is easily verified that this constitutes a linear function family. Due to space limitation, the proofs of the following two lemmas are postponed to the full version [34].

Lemma 3. *Assuming that the* DDH *assumption holds relative to* GGen*, the linear function family* $\mathsf{LF}_{\mathsf{DDH}}$ *is* ε_{l}*-lossiness admitting, with* $\varepsilon_{\mathsf{l}} \leq 3/p$*. Concretely, for any PPT algorithm* $\mathcal{A}$ *there is a PPT algorithm* $\mathcal{B}$ *with* $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{A})$ *and*

$$\mathsf{Adv}^{\mathsf{keydist}}_{\mathcal{A},\mathsf{LF}_{\mathsf{DDH}}}(\lambda) \leq \mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{B},\mathsf{GGen}}(\lambda).$$

Lemma 4. *Linear function family* $\mathsf{LF}_{\mathsf{DDH}}$ *satisfies* $\varepsilon_{\mathsf{al}}$*-aggregation lossy soundness with* $\varepsilon_{\mathsf{al}} \leq 4/p$*.*

[8] We omit the description of $\mathbb{G}$ from par to make the presentation concise.

4.2 Commitment Scheme

We give a special trapdoor commitment scheme $\mathsf{CMT}_{\mathsf{DDH}} = (\mathsf{BGen}, \mathsf{TGen}, \mathsf{Com}, \mathsf{TCom}, \mathsf{TCol})$ for the linear function family $\mathsf{LF}_{\mathsf{DDH}}$. For given parameters of $\mathsf{LF}_{\mathsf{DDH}}$, the commitment scheme has key space $\mathcal{K} := \mathbb{G}^{3\times 3}$ and message space $\mathcal{D} = \mathbb{G} \times \mathbb{G}$. It has randomness space $\mathcal{G} = \mathbb{Z}_p^3$ and commitment space $\mathcal{H} = \mathbb{G}^3$. Both are associated with the natural componentwise group operations. We describe the algorithms of the scheme verbally.

- $\mathsf{BGen}(\mathsf{par}) \to \mathsf{ck}$: Sample $g_1, g_2, g_3 \xleftarrow{\$} \mathbb{G}$, and $a, b \xleftarrow{\$} \mathbb{Z}_p$, and set

$$\mathsf{ck} := \mathbf{A} := \begin{pmatrix} A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix} := \begin{pmatrix} g_1 & g_1^a & g_1^b \\ g_2 & g_2^a & g_2^b \\ g_3 & g_3^a & g_3^b \end{pmatrix} \in \mathbb{G}^{3\times 3}.$$

- $\mathsf{TGen}(\mathsf{par}, X = (X_1, X_2)) \to (\mathsf{ck}, \mathsf{td})$: Sample $d_{i,j} \xleftarrow{\$} \mathbb{Z}_p$ for all $(i,j) \in [3] \times [3]$ and set

$$\mathsf{ck} := \mathbf{A} := \begin{pmatrix} A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix} := \begin{pmatrix} g^{d_{1,1}} & g^{d_{1,2}} & g^{d_{1,3}} \\ X_1^{d_{2,1}} & X_1^{d_{2,2}} & X_1^{d_{2,3}} \\ X_2^{d_{3,1}} & X_2^{d_{3,2}} & X_2^{d_{3,3}} \end{pmatrix} \in \mathbb{G}^{3\times 3}.$$

  Next, set

$$\mathsf{td} := (\mathbf{D}, X_1, X_2), \text{ for } \mathbf{D} := \begin{pmatrix} d_{1,1} & d_{1,2} & d_{1,3} \\ d_{2,1} & d_{2,2} & d_{2,3} \\ d_{3,1} & d_{3,2} & d_{3,3} \end{pmatrix} \in \mathbb{Z}_p^{3\times 3}.$$

- $\mathsf{Com}(\mathsf{ck}, R = (R_1, R_2); \varphi) \to \mathsf{com}$: Let $\varphi = (\alpha, \beta, \gamma) \in \mathbb{Z}_p^3$. Compute

$$\mathsf{com} := (C_0, C_1, C_2), \text{ for } \begin{pmatrix} C_0 \\ C_1 \\ C_2 \end{pmatrix} := \begin{pmatrix} A_{1,1}^{\alpha} \cdot A_{1,2}^{\beta} \cdot A_{1,3}^{\gamma} \\ R_1 \cdot A_{2,1}^{\alpha} \cdot A_{2,2}^{\beta} \cdot A_{2,3}^{\gamma} \\ R_2 \cdot A_{3,1}^{\alpha} \cdot A_{3,2}^{\beta} \cdot A_{3,3}^{\gamma} \end{pmatrix}.$$

- $\mathsf{TCom}(\mathsf{ck}, \mathsf{td}) \to (\mathsf{com}, St)$: Sample $\tau, \rho_1, \rho_2, s \xleftarrow{\$} \mathbb{Z}_p$. Set $St := (\mathsf{td}, \tau, \rho_1, \rho_2, s)$ and compute

$$\mathsf{com} := (C_0, C_1, C_2), \text{ for } \begin{pmatrix} C_0 \\ C_1 \\ C_2 \end{pmatrix} := \begin{pmatrix} g^{\tau} \\ X_1^{\rho_1} \cdot g^s \\ X_2^{\rho_2} \cdot h^s \end{pmatrix}.$$

- $\mathsf{TCol}(St, c) \to (\varphi, R, s)$: Set $R := (R_1, R_2) := \left(g^s \cdot X_1^{-c}, h^s \cdot X_2^{-c}\right)$. Then, if $\mathbf{D}$ is not invertible, return $\perp$. Otherwise, compute

$$\varphi := (\alpha, \beta, \gamma), \text{ for } \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} = \mathbf{D}^{-1} \cdot \begin{pmatrix} \tau \\ \rho_1 + c \\ \rho_2 + c \end{pmatrix}.$$

Theorem 3. *If the* DDH *assumption holds relative to* GGen*, then* $\mathsf{CMT}_{\mathsf{DDH}}$ *is a* $(\varepsilon_{\mathsf{b}}, \varepsilon_{\mathsf{g}}, \varepsilon_{\mathsf{t}})$*-special commitment scheme for* $\mathsf{LF}_{\mathsf{DDH}}$*, with*

$$\varepsilon_{\mathsf{b}} \leq 1/p, \quad \varepsilon_{\mathsf{g}} \leq 2/p, \quad \varepsilon_{\mathsf{t}} \leq 6/p.$$

Concretely, for any PPT algorithm $\mathcal{A}$*, there is a PPT algorithm* $\mathcal{B}$ *with* $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{A})$ *and*

$$\mathsf{Adv}^{Q\text{-}\mathsf{keydist}}_{\mathcal{A},\mathsf{CMT}_{\mathsf{DDH}}}(\lambda) \leq \mathsf{Adv}^{\mathsf{uDDH3}}_{\mathcal{B},\mathsf{GGen}}(\lambda) + \frac{6}{p}.$$

The homomorphism property is trivial to check. Next, we define the set Good as in the definition of a special commitment scheme. Namely, we define

$$\mathsf{Good} = \{((g,h),x) \in \mathbb{G}^2 \times \mathbb{Z}_p \mid (g,h) \in \mathsf{LF.Gen}(1^\lambda) \wedge h \neq g^0 \wedge x \neq 0\}.$$

Clearly, for $(g,h) \leftarrow \mathsf{LF.Gen}(1^\lambda)$ and $x \xleftarrow{\$} \mathbb{Z}_p$, the probability that $((g,h),x) \notin \mathsf{Good}$ is at most $2/p$. Therefore, $\varepsilon_{\mathsf{g}} \leq 2/p$. In the following we also need the following observation: If $((g,h),x) \in \mathsf{Good}$, then the elements g, h, g^x, h^x are all generators of $\mathbb{G}$. The rest of proof of the theorem is given in separate lemmas.

Lemma 5. $\mathsf{CMT}_{\mathsf{DDH}}$ *satisfies the uniform keys property of an* $(\varepsilon_{\mathsf{b}}, \varepsilon_{\mathsf{g}}, \varepsilon_{\mathsf{t}})$*-special commitment scheme for* $\mathsf{LF}_{\mathsf{DDH}}$*.*

Proof. Let $(\mathsf{par}, x) \in \mathsf{Good}$ for $\mathsf{par} = (g,h)$. Let $(X_1, X_2) = \mathsf{F}(x) = (g^x, h^x)$. Consider the distribution of ck for $(\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{TGen}(\mathsf{par}, (X_1, X_2))$. Then ck has the form

$$\begin{pmatrix} g^{d_{1,1}} & g^{d_{1,2}} & g^{d_{1,3}} \\ X_1^{d_{2,1}} & X_1^{d_{2,2}} & X_1^{d_{2,3}} \\ X_2^{d_{3,1}} & X_2^{d_{3,2}} & X_2^{d_{3,3}} \end{pmatrix} \in \mathbb{G}^{3\times 3}$$

for uniformly random and independent exponents $d_{i,j} \in \mathbb{Z}_p$ $(i,j \in [3])$. As g, X_1, X_2 are generators, we see that ck is uniform over $\mathbb{G}^{3\times 3}$, proving the claim. □

Lemma 6. $\mathsf{CMT}_{\mathsf{DDH}}$ *satisfies the special trapdoor property of an* $(\varepsilon_{\mathsf{b}}, \varepsilon_{\mathsf{g}}, \varepsilon_{\mathsf{t}})$*-special commitment scheme for* $\mathsf{LF}_{\mathsf{DDH}}$*, where* $\varepsilon_{\mathsf{t}} \leq 6/p$*.*

Proof. Let $((g,h),x) \in \mathsf{Good}$ and $c \in \mathbb{Z}_p$. Set $(X_1, X_2) := (g^x, h^x)$. We have to show that the distributions $\mathcal{T}_0$ and $\mathcal{T}_1$ of tuples

$$((g,h), \mathbf{A}, \mathbf{D}, X_1, X_2, x, c, (C_0, C_1, C_2), \alpha, \beta, \gamma, R_1, R_2, s)$$

are identical. Here, we have $(\mathbf{A}, \mathbf{D}, X_1, X_2) \leftarrow \mathsf{TGen}(\mathsf{par}, (X_1, X_2))$. The remaining components in $\mathcal{T}_0$ are generated via

$$((C_0, C_1, C_2), St) \leftarrow \mathsf{TCom}(\mathsf{ck}, \mathsf{td}), \ ((\alpha, \beta, \gamma), (R_1, R_2), s) \leftarrow \mathsf{TCol}(St, c),$$

and in $\mathcal{T}_1$ via

$$r \xleftarrow{\$} \mathbb{Z}_p, \ R_1 := g^r, \ R_2 := h^r, \ s := c \cdot x + r$$
$$\alpha, \beta, \gamma \xleftarrow{\$} \mathbb{Z}_p, \ (C_0, C_1, C_2) := \mathsf{Com}(\mathbf{A}, (R_1, R_2); (\alpha, \beta, \gamma)).$$

First, we make the assumption that in both distributions, the matrix $\mathbf{D}$ has full rank. The probability that this does not hold can easily be bounded by $3/p$.

We can equivalently[9] write $\mathcal{T}_1$ as

$$s \xleftarrow{\$} \mathbb{Z}_p,\ R_1 := g^s \cdot X_1^{-c},\ R_2 := h^s \cdot X_2^{-c},$$
$$\alpha, \beta, \gamma \xleftarrow{\$} \mathbb{Z}_p,\ (C_0, C_1, C_2) := \mathsf{Com}(\mathbf{A}, (R_1, R_2); (\alpha, \beta, \gamma)).$$

Using that $\mathbf{D}$ is full rank and g, X_1, X_2 are generators of $\mathbb{G}$, we see that in this distribution, (C_0, C_1, C_2) is uniform over $\mathbb{G}^3$. Therefore, this is identically distributed to the distribution that we get from

$$s \xleftarrow{\$} \mathbb{Z}_p,\ R_1 := g^s \cdot X_1^{-c},\ R_2 := h^s \cdot X_2^{-c},$$
$$\tau, \rho_1, \rho_2 \xleftarrow{\$} \mathbb{Z}_p,\ (C_0, C_1, C_2) := (g^\tau, X_1^{\rho_1} g^s, X_2^{\rho_2} h^s),$$

and then finding the unique values (α, β, γ) that satisfy $(C_0, C_1, C_2) = \mathsf{Com}(\mathbf{A}, (R_1, R_2); (\alpha, \beta, \gamma))$. We claim that this can be done using $(\alpha, \beta, \gamma)^t := \mathbf{D}^{-1}(\tau, \rho_1 + c, \rho_2 + c)^t$, which is equivalent to distribution $\mathcal{T}_0$.

To see this, note that $(C_0, C_1, C_2) = \mathsf{Com}(\mathbf{A}, (R_1, R_2); (\alpha, \beta, \gamma))$ is equivalent to

$$\begin{pmatrix} C_0 \\ C_1 \\ C_2 \end{pmatrix} = \begin{pmatrix} A_{1,1}^{\alpha} \cdot A_{1,2}^{\beta} \cdot A_{1,3}^{\gamma} \\ R_1 \cdot A_{2,1}^{\alpha} \cdot A_{2,2}^{\beta} \cdot A_{2,3}^{\gamma} \\ R_1 \cdot A_{3,1}^{\alpha} \cdot A_{3,2}^{\beta} \cdot A_{3,3}^{\gamma} \end{pmatrix} = \begin{pmatrix} g^{d_{1,1}\alpha} \cdot g^{d_{1,2}\beta} \cdot g^{d_{1,3}\gamma} \\ g^s \cdot X_1^{-c} \cdot X_1^{d_{2,1}\alpha} \cdot X_1^{d_{2,2}\beta} \cdot X_1^{d_{2,3}\gamma} \\ h^s \cdot X_2^{-c} \cdot X_2^{d_{3,1}\alpha} \cdot X_2^{d_{3,2}\beta} \cdot X_2^{d_{3,3}\gamma} \end{pmatrix}.$$

Using the way we generate (C_0, C_1, C_2), we see that the g^s and h^s terms cancel out, and this is equivalent to

$$\begin{pmatrix} g^\tau \\ X_1^{\rho_1} \\ X_2^{\rho_2} \end{pmatrix} = \begin{pmatrix} g^{d_{1,1}\alpha} \cdot g^{d_{1,2}\beta} \cdot g^{d_{1,3}\gamma} \\ X_1^{d_{2,1}\alpha} \cdot X_1^{d_{2,2}\beta} \cdot X_1^{d_{2,3}\gamma} \\ X_2^{d_{3,1}\alpha} \cdot X_2^{d_{3,2}\beta} \cdot X_2^{d_{3,3}\gamma} \end{pmatrix} \iff \begin{pmatrix} \tau \\ \rho_1 + c \\ \rho_2 + c \end{pmatrix} = \mathbf{D} \cdot \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix}.$$

This concludes the proof. □

Lemma 7. $\mathsf{CMT}_{\mathsf{DDH}}$ *satisfies the statistically binding property of an* $(\varepsilon_{\mathsf{b}}, \varepsilon_{\mathsf{g}}, \varepsilon_{\mathsf{t}})$*-special commitment scheme for* $\mathsf{LF}_{\mathsf{DDH}}$*, with* $\varepsilon_{\mathsf{b}} \leq 1/p$.

Proof. We describe an unbounded algorithm Ext, that takes as input a commitment key $\mathsf{ck} = \mathbf{A} = (A_{i,j})_{i,j} \in \mathbb{G}^{3\times 3}$, and a commitment $\mathsf{com} = (C_0, C_1, C_2) \in \mathbb{G}^3$, and outputs a tuple $R = (R_1, R_2) \in \mathbb{G} \times \mathbb{G}$. It is given as follows:

1. Extract discrete logarithms $\mathbf{c} = (c_0, c_1, c_2)^t \in \mathbb{Z}_p^3$ and $\mathbf{a} = (a_0, a_1, a_2)^t \in \mathbb{Z}_p^3$ such that
$$\begin{pmatrix} C_0 \\ C_1 \\ C_2 \end{pmatrix} = \begin{pmatrix} g^{c_0} \\ g^{c_1} \\ g^{c_2} \end{pmatrix} \text{ and } \begin{pmatrix} A_{1,1} \\ A_{2,1} \\ A_{3,1} \end{pmatrix} = \begin{pmatrix} g^{a_0} \\ g^{a_1} \\ g^{a_2} \end{pmatrix}.$$
2. If $a_0 = 0$, return $\perp$. Otherwise, let $\mathbf{e}_2 = (0, 1, 0)^t$ and $\mathbf{e}_3 = (0, 0, 1)^t$. Note that $\mathbf{a}, \mathbf{e}_2, \mathbf{e}_3$ form a basis of $\mathbb{Z}_p^3$.

[9] This corresponds to the HVZK property of linear identification protocols.

3. Write $\mathbf{c}$ as $\mathbf{c} = t\mathbf{a} + r_1\mathbf{e}_2 + r_2\mathbf{e}_3$ for $t, r_1, r_2 \in \mathbb{Z}_p$, and return $(R_1, R_2) := (g^{r_1}, g^{r_2})$.

To finish the proof, let $\mathcal{A}$ be any algorithm. We have to bound the probability

$$\Pr\left[\begin{array}{l|l} \mathsf{Com}(\mathbf{A}, (R_1', R_2'); \varphi') = (C_0, C_1, C_2) & (g,h) \leftarrow \mathsf{LF.Gen}(1^\lambda), \\ \qquad \wedge \ (R_1, R_2) \neq (R_1', R_2') & \mathbf{A} \leftarrow \mathsf{BGen}(\mathsf{par}), \\ & ((C_0, C_1, C_2), St) \leftarrow \mathcal{A}(\mathbf{A}), \\ & (R_1, R_2) \leftarrow \mathsf{Ext}(\mathbf{A}, (C_0, C_1, C_2)), \\ & (R_1, R_2', \varphi') \leftarrow \mathcal{A}(St) \end{array}\right].$$

Note that the probability that Ext outputs $\perp$ in this experiment is $1/p$, as $A_{1,1}$ is uniform in $\mathbb{G}$. We assume that Ext does not output $\perp$, and want to show that the above probability conditioned on this event is zero. First, it is easy to see that we have $\mathsf{Com}(\mathbf{A}, (R_1, R_2); (t, 0, 0)) = (C_0, C_1, C_2)$. Further, assume that $\mathcal{A}$ outputs $(R_1', R_2') = (g^{r_1'}, g^{r_2'})$ and $\varphi' = (\alpha, \beta, \gamma)$ such that

$$\mathsf{Com}(\mathbf{A}, (R_1', R_2'); \varphi') = (C_0, C_1, C_2) = \mathsf{Com}(\mathbf{A}, (R_1, R_2); (t, 0, 0)).$$

Using the definition of Com and BGen, we see that this implies the vector $(0, r_1 - r_1', r_2 - r_2')^t$ is in the span of $\mathbf{a}$. As $a_0 \neq 0$ this implies that it is the zero vector, showing that $R_1 = R_1'$ and $R_2 = R_2'$. □

Lemma 8. *For any PPT algorithm $\mathcal{A}$, there is a PPT algorithm $\mathcal{B}$ with $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{A})$ and*

$$\mathsf{Adv}^{Q\text{-}\mathsf{keydist}}_{\mathcal{A},\mathsf{CMT}_{\mathsf{DDH}}}(\lambda) \leq \mathsf{Adv}^{\mathsf{uDDH3}}_{\mathcal{B},\mathsf{GGen}}(\lambda) + \frac{6}{p}.$$

The lemma is proven by a simple reduction. Looking at one fixed commitment key $\mathbf{A}_i$, indistinguishability would directly follow from the $\mathsf{uDDH3}$ assumption. To give a tight reduction for any $Q = \mathsf{poly}(\lambda)$, we use the random self-reducibility of $\mathsf{uDDH3}$. We postpone it to the full version [34].

4.3 Efficiency

In our full version [34], we discuss the efficiency of our schemes both asymptotically, as well as in terms of concrete parameters.

Acknowledgments. We thank the anonymous reviewers from Eurocrypt 2023 for their useful feedback and suggestions. In particular, it was pointed out the similarity between the commitment scheme of Bagherzandi, Cheon, and Jarecki in [3] and ours.

References

1. Abdalla, M., Fouque, P.A., Lyubashevsky, V., Tibouchi, M.: Tightly-secure signatures from lossy identification schemes. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 572–590. Springer, Heidelberg (Apr 2012). https://doi.org/10.1007/978-3-642-29011-4_34

2. Alper, H.K., Burdges, J.: Two-round trip schnorr multi-signatures via delinearized witnesses. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part I. LNCS, vol. 12825, pp. 157–188. Springer, Heidelberg, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84242-0_7
3. Bagherzandi, A., Cheon, J.H., Jarecki, S.: Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. In: Ning, P., Syverson, P.F., Jha, S. (eds.) ACM CCS 2008. pp. 449–458. ACM Press (Oct 2008). https://doi.org/10.1145/1455770.1455827
4. Bellare, M., Dai, W.: Chain reductions for multi-signatures and the HBMS scheme. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part IV. LNCS, vol. 13093, pp. 650–678. Springer, Heidelberg (Dec 2021). https://doi.org/10.1007/978-3-030-92068-5_22
5. Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: Juels, A., Wright, R.N., De Capitani di Vimercati, S. (eds.) ACM CCS 2006. pp. 390–399. ACM Press (Oct / Nov 2006). https://doi.org/10.1145/1180405.1180453
6. Blazy, O., Kiltz, E., Pan, J.: (Hierarchical) identity-based encryption from affine message authentication. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 408–425. Springer, Heidelberg (Aug 2014). https://doi.org/10.1007/978-3-662-44371-2_23
7. Boldyreva, A.: Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In: Desmedt, Y. (ed.) PKC 2003. LNCS, vol. 2567, pp. 31–46. Springer, Heidelberg (Jan 2003). https://doi.org/10.1007/3-540-36288-6_3
8. Boneh, D., Drijvers, M., Neven, G.: Compact multi-signatures for smaller blockchains. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part II. LNCS, vol. 11273, pp. 435–464. Springer, Heidelberg (Dec 2018). https://doi.org/10.1007/978-3-030-03329-3_15
9. Boschini, C., Takahashi, A., Tibouchi, M.: MuSig-L: Lattice-based multi-signature with single-round online phase. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part II. LNCS, vol. 13508, pp. 276–305. Springer, Heidelberg (Aug 2022). https://doi.org/10.1007/978-3-031-15979-4_10
10. Chairattana-Apirom, R., Hanzlik, L., Loss, J., Lysyanskaya, A., Wagner, B.: PI-cut-choo and friends: Compact blind signatures via parallel instance cut-and-choose and more. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part III. LNCS, vol. 13509, pp. 3–31. Springer, Heidelberg (Aug 2022). https://doi.org/10.1007/978-3-031-15982-4_1
11. Crites, E., Komlo, C., Maller, M.: How to prove schnorr assuming schnorr: Security of multi- and threshold signatures. Cryptology ePrint Archive, Report 2021/1375 (2021), https://eprint.iacr.org/2021/1375
12. Damgård, I.: Efficient concurrent zero-knowledge in the auxiliary string model. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 418–430. Springer, Heidelberg (May 2000). https://doi.org/10.1007/3-540-45539-6_30
13. Damgård, I., Orlandi, C., Takahashi, A., Tibouchi, M.: Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. In: Garay, J. (ed.) PKC 2021, Part I. LNCS, vol. 12710, pp. 99–130. Springer, Heidelberg (May 2021). https://doi.org/10.1007/978-3-030-75245-3_5
14. Drijvers, M., Edalatnejad, K., Ford, B., Kiltz, E., Loss, J., Neven, G., Stepanovs, I.: On the security of two-round multi-signatures. In: 2019 IEEE Symposium on Security and Privacy. pp. 1084–1101. IEEE Computer Society Press (May 2019). https://doi.org/10.1109/SP.2019.00050

15. Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.: An algebraic framework for Diffie-Hellman assumptions. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 129–147. Springer, Heidelberg (Aug 2013). https://doi.org/10.1007/978-3-642-40084-1_8
16. Fukumitsu, M., Hasegawa, S.: A tightly secure ddh-based multisignature with public-key aggregation. Int. J. Netw. Comput. **11**(2), 319–337 (2021), http://www.ijnc.org/index.php/ijnc/article/view/257
17. Goh, E.J., Jarecki, S., Katz, J., Wang, N.: Efficient signature schemes with tight reductions to the Diffie-Hellman problems. Journal of Cryptology **20**(4), 493–514 (Oct 2007). https://doi.org/10.1007/s00145-007-0549-3
18. Groth, J.: Homomorphic trapdoor commitments to group elements. Cryptology ePrint Archive, Report 2009/007 (2009), https://eprint.iacr.org/2009/007
19. Groth, J., Ostrovsky, R., Sahai, A.: Non-interactive zaps and new techniques for NIZK. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 97–111. Springer, Heidelberg (Aug 2006). https://doi.org/10.1007/11818175_6
20. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (Apr 2008). https://doi.org/10.1007/978-3-540-78967-3_24
21. Guillou, L.C., Quisquater, J.J.: A practical zero-knowledge protocol fitted to security microprocessor minimizing both trasmission and memory. In: Günther, C.G. (ed.) EUROCRYPT'88. LNCS, vol. 330, pp. 123–128. Springer, Heidelberg (May 1988). https://doi.org/10.1007/3-540-45961-8_11
22. Han, S., Jager, T., Kiltz, E., Liu, S., Pan, J., Riepel, D., Schäge, S.: Authenticated key exchange and signatures with tight security in the standard model. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part IV. LNCS, vol. 12828, pp. 670–700. Springer, Heidelberg, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84259-8_23
23. Hauck, E., Kiltz, E., Loss, J.: A modular treatment of blind signatures from identification schemes. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part III. LNCS, vol. 11478, pp. 345–375. Springer, Heidelberg (May 2019). https://doi.org/10.1007/978-3-030-17659-4_12
24. Itakura, K., Nakamura, K.: A public-key cryptosystem suitable for digital multisignatures. NEC Research & Development (71), 1–8 (1983)
25. Katz, J., Loss, J., Rosenberg, M.: Boosting the security of blind signature schemes. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part IV. LNCS, vol. 13093, pp. 468–492. Springer, Heidelberg (Dec 2021). https://doi.org/10.1007/978-3-030-92068-5_16
26. Katz, J., Wang, N.: Efficiency improvements for signature schemes with tight security reductions. In: Jajodia, S., Atluri, V., Jaeger, T. (eds.) ACM CCS 2003. pp. 155–164. ACM Press (Oct 2003). https://doi.org/10.1145/948109.948132
27. Kiltz, E., Masny, D., Pan, J.: Optimal security proofs for signatures from identification schemes. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part II. LNCS, vol. 9815, pp. 33–61. Springer, Heidelberg (Aug 2016). https://doi.org/10.1007/978-3-662-53008-5_2
28. Langrehr, R., Pan, J.: Unbounded HIBE with tight security. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part II. LNCS, vol. 12492, pp. 129–159. Springer, Heidelberg (Dec 2020). https://doi.org/10.1007/978-3-030-64834-3_5
29. Lu, S., Ostrovsky, R., Sahai, A., Shacham, H., Waters, B.: Sequential aggregate signatures and multisignatures without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 465–485. Springer, Heidelberg (May / Jun 2006). https://doi.org/10.1007/11761679_28

30. Maxwell, G., Poelstra, A., Seurin, Y., Wuille, P.: Simple schnorr multi-signatures with applications to bitcoin. Des. Codes Cryptogr. **87**(9), 2139–2164 (2019). https://doi.org/10.1007/s10623-019-00608-x, https://doi.org/10.1007/s10623-019-00608-x
31. Micali, S., Ohta, K., Reyzin, L.: Accountable-subgroup multisignatures: Extended abstract. In: Reiter, M.K., Samarati, P. (eds.) ACM CCS 2001. pp. 245–254. ACM Press (Nov 2001). https://doi.org/10.1145/501983.502017
32. Nick, J., Ruffing, T., Seurin, Y.: MuSig2: Simple two-round Schnorr multi-signatures. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part I. LNCS, vol. 12825, pp. 189–221. Springer, Heidelberg, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84242-0_8
33. Nick, J., Ruffing, T., Seurin, Y., Wuille, P.: MuSig-DN: Schnorr multi-signatures with verifiably deterministic nonces. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) ACM CCS 2020. pp. 1717–1731. ACM Press (Nov 2020). https://doi.org/10.1145/3372297.3417236
34. Pan, J., Wagner, B.: Chopsticks: Fork-free two-round multi-signatures from non-interactive assumptions. Cryptology ePrint Archive, Paper 2023/198 (2023), https://eprint.iacr.org/2023/198, https://eprint.iacr.org/2023/198
35. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO'91. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (Aug 1992). https://doi.org/10.1007/3-540-46766-1_9
36. Schnorr, C.P.: Efficient signature generation by smart cards. Journal of Cryptology **4**(3), 161–174 (Jan 1991). https://doi.org/10.1007/BF00196725
37. Tessaro, S., Zhu, C.: Threshold and multi-signature schemes from linear hash functions. In: Eurocrypt 2023 (to appear). LNCS, Springer, Heidelberg (2023)